ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΥΠΟΛΟΓΙΣΤΩΝ

Το Πρωτόκολλο Universal Serial Bus 1.1 (USB 1.1) για Προσωπικούς Υπολογιστές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΠΟΣΤΟΛΟΣ Π. ΚΑΤΑΛΑΓΑΡΓΙΑΝΟΣ

Επιβλέπων : ΓΕΩΡΓΙΟΣ Κ. ΠΑΠΑΚΩΝΣΤΑΝΤΙΝΟΥ
Καθηγητής ΕΜΠ

Αθήνα, Ιανουάριος 2006

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ
ΥΠΟΛΟΓΙΣΤΩΝ

Το Πρωτόκολλο Universal Serial Bus 1.1 (USB 1.1) για Προσωπικούς Υπολογιστές

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΠΟΣΤΟΛΟΣ Π. ΚΑΤΑΛΑΓΑΡΓΙΑΝΟΣ

Επιβλέπων : ΓΕΩΡΓΙΟΣ Κ. ΠΑΠΑΚΩΝΣΤΑΝΤΙΝΟΥ
Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

.....	.....	.....
Γ. ΠΑΠΑΚΩΝΣΤΑΝΤΙΝΟΥ	Π. ΤΣΑΝΑΚΑΣ	Ν. ΚΟΖΥΡΗΣ
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Επίκουρος Καθηγητής ΕΜΠ

Αθήνα, Ιανουάριος 2006

.....

ΑΠΟΣΤΟΛΟΣ Π. ΚΑΤΑΛΑΓΑΡΓΙΑΝΟΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ΠΕΡΙΛΗΨΗ

Ο σκοπός της διπλωματικής εργασίας είναι η ανάπτυξη ενός μέρους του λογισμικού οδήγησης του ελεγκτή USB ενός κεντρικού υπολογιστικού συστήματος (USB Host) στο οποίο προσαρτώνται συσκευές μέσω ενός διαύλου Universal Serial Bus 1.1 (USB 1.1). Το λογισμικό συγγράφτηκε για το λειτουργικό σύστημα MS-DOS και για ένα ελεγκτή του USB Host που είναι συμβατός με τις προδιαγραφές του προτύπου Universal Host Controller Interface (UHCI), το οποίο αποτελεί την πιο δημοφιλή τυποποίηση διαπροσωπείας (Interface) υλικού και λογισμικού του USB 1.1 μέσω της οποίας επιτυγχάνεται η επικοινωνία μεταξύ του υλικού του ελεγκτή και του λογισμικού οδήγησής του.

Στα πλαίσια της εργασίας πραγματοποιείται μια λεπτομερής ανάλυση των προδιαγραφών του Universal Serial Bus 1.1 και του Universal Host Controller Interface. Σε ό,τι αφορά τις προδιαγραφές του USB 1.1, η ανάλυση εστιάζεται στην παρουσίαση των αρμοδιοτήτων κάθε συνιστώσας του USB Host και του τρόπου με τον οποίο επιτυγχάνεται η επικοινωνία μεταξύ των διαφόρων συνιστωσών, στους τύπους μεταφοράς δεδομένων που υποστηρίζονται από το USB, στις απαιτήσεις του προτύπου σε ό,τι αφορά την χρονοδρομολόγηση των αιτήσεων μεταφοράς δεδομένων, και στο πρωτόκολλο διαύλου του USB 1.1, δηλαδή το πρωτόκολλο βάσει του οποίου πραγματοποείται η μεταφορά δεδομένων στον δίαυλο. Η περιγραφή του UHCI περιλαμβάνει την παρουσίαση των καταχωρητών της διαπροσωπείας, των δομών δεδομένων και των πληροφοριών που περιέχουν μέσω των οποίων το λογισμικό παραδίδει στο υλικό το χρονοδιάγραμμα (Schedule) των αιτήσεων μεταφοράς δεδομένων προς εκτέλεση, της μεθόδου βάσει της οποίας το υλικό εκτελεί το χρονοδιάγραμμα, και του συνόλου των διακοπών που εγείρονται από το υλικό.

Το λογισμικό που αναπτύχθηκε τηρεί τις παραπάνω προδιαγραφές και υλοποιεί την χρονοδρομολόγηση των αιτήσεων μεταφοράς δεδομένων και την παράδοσή τους σε μορφή αναγνώσιμη από το υλικό, καθώς και τις δομές δεδομένων για την περιγραφή των χαρακτηριστικών κάθε μεταφοράς δεδομένων και την επικοινωνία με τις υπόλοιπες συνιστώσες του συστήματος λογισμικού του USB Host. Η προσθήκη συναρτήσεων που θα διαχειρίζονται τις διακοπές και τα γεγονότα που αναφέρονται από τους καταχωρητές της διαπροσωπείας θα έχει ως αποτέλεσμα ένα ολοκληρωμένο σύστημα λογισμικού οδήγησης ενός UHCI ελεγκτή του Host για το λειτουργικό σύστημα MS-DOS.

Λέξεις Κλειδιά

Universal Serial Bus 1.1, USB 1.1, Universal Host Controller Interface, UHCI, USB Host, Ελεγκτής του Host, Λογισμικό Οδήγησης του Ελεγκτή του Host, Χρονοδρομολόγηση, Αιτήσεις Μεταφοράς Δεδομένων, Μεταφορά Δεδομένων, MS-DOS, Περιφερειακές Συσκευές, Επικοινωνία, Δίαυλος.

DIPLOMA THESIS TITLE

The Universal Serial Bus 1.1 (USB 1.1) Protocol for Personal Computers

ABSTRACT

The scope of this diploma thesis is the development of a part of the software driver for the USB controller of a USB Host to which several devices are attached via a Universal Serial Bus (USB 1.1). The software was written for the MS-DOS operating system and for a host controller that is compatible with the specifications of the Universal Host Controller Interface (UHCI) model, which is the most popular model of interface between the hardware and the software of USB 1.1, via which communication between hardware and its software driver is achieved.

Within the framework of this thesis a detailed description of the Universal Serial Bus 1.1 and the Universal Host Controller Interface specifications is carried out. With regard to the USB 1.1 specifications, the analysis focuses on the presentation of the responsibilities of each component of the USB Host and on the way that communation is achieved between these components, the data transfer types that are supported by USB, the requirements of the model regarding the scheduling of the data trasfer requests, and the bus protocol of USB 1.1 which describes the way that data transfers are carried out on the bus. The description of UHCI includes the presentation of the interface registers, the data strustures and the information they carry which are used by the software in order to deliver to the hardware the schedule of data transfer requests to be executed, the method which is followd by the hardware in order to execute the schedule, and the set of interrupts that are generated by hardware.

The developed software meets these specifications and implements the processes of the scheduling for the data transfer requests and their delivery in a comprehensive format to the hardware and the data structures which describe the attributes of each data transfer and provide communication with other components of the software system of the USB Host. The addition of functions that handle the hardware interrupts and the events that are reported by the interface registers will result to a complete software driver for a UHCI Host Controller for the MS-DOS operating system.

Key Words

Universal Serial Bus 1.1, USB 1.1, Universal Host Controller Interface, UHCI, USB Host, Host Controller, Host Controller Driver, Scheduling, Data Transfer Requests, Data Transfer, MS-DOS, Peripherals, Communication, Bus.

ΠΕΡΙΕΧΟΜΕΝΑ

ΣΧΗΜΑΤΑ

ΠΙΝΑΚΕΣ

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

1.1 Χαρακτηριστικά του Universal Serial Bus

Το Universal Serial Bus 1.1 (*USB 1.1*) είναι ένας Εξωτερικός Δίαυλος *(External Bus)*, σχεδιασμένος για να καλύψει τις ανάγκες επικοινωνίας των προσωπικών υπολογιστών *(Personal Computers, PCs)* με περιφερειακές συσκευές σύγχρονης τεχνολογίας. Οι προδιαγραφές του USB 1.1 τέθηκαν το 1998 από τις εταιρίες Compaq, Intel, Microsoft, και NEC, με στόχο την ικανοποίηση των παρακάτω κριτηριών:

- Ευκολία στην προσάρτηση περιφερειακών μονάδων σε ένα υπολογιστικό σύστημα μέσω του USB.
- Ελαχιστοποίηση του κόστους υλοποίησης, υποστηρίζοντας ρυθμούς μεταφοράς δεδομένων μέχρι 12 Mb/s.
- Πλήρης υποστήριξη μεταφοράς δεδομένων πραγματικού χρόνου (φωνής, ήχου, συμπιεσμένης εικόνας).
- Υποστήριξη εγγυημένης μεταφοράς δεδομένων σε ασύγχρονες *(Asynchronous)* μεταφορές δεδομένων, αλλά και ισόχρονων *(Isochronous)* μεταφορών δεδομένων (χωρίς ανίχνευση και διόρθωση λαθών).
- Συμβατότητα με τα υπάρχοντα υπολογιστικά συστήματα.
- Υποστήριξη των απαιτήσεων των υπάρχοντων περιφερειακών συσκευών, αλλά με στόχο την υποστήριξη νέων τύπων συσκευών αυξημένων απαιτήσεων εύρους ζώνης.

Στον Πίνακα 1.1.1 παρουσιάζεται η ταξινόμηση των διάφορων περιφερειακών συσκευών με βάση την ταχύτητα μεταφοράς δεδομένων που απαιτεί κάθε συσκευή. Το USB 1.1 υποστηρίζει συσκευές χαμηλής *(Low-Speed)* και μεσαίας ταχύτητας (*Medium-Speed)*. Αντίθετα, το USB 1.1, του οποίου η μέγιστη δυνατή ταχύτητα μεταφοράς δεδομένων περιορίζεται στα 12 Mb/s, είναι ακατάλληλο για υποστήριξη συσκευών υψηλής ταχύτητας *(High Speed)*.

Πίνακας 1.1.1: Ταξινόμηση Συσκευών με βάση την Ταχύτητα Μεταφοράς Δεδομένων

Επιδόσεις	Εφαρμογές	Ιδιότητες
Χαμηλής Ταχύτητας 10 – 100 Kb/s	**Διαλογικές Συσκευές** Πληκτρολόγια, Ποντίκια Joysticks, Joypads Περιφερειακά Εικονικής Πραγματικότητας (Virtual Reality)	Χαμηλό κόστος Άμεση σύνδεση- αποσύνδεση Ευκολία χρήσης Πολλαπλά περιφερειακά
Μεσαίας Ταχύτητας	**Τηλεφωνία, Ήχος, Συμπιεσμένη Εικόνα**	Χαμηλό κόστος Ευκολία χρήσης

500 Kb/s – 10 Mb/s	ISDN PBX POTS Συσκευές Audio	Εγγυημένη καθυστέρηση Εγγυημένο εύρος ζώνης Δυναμική σύνδεση- αποσύνδεση Πολλαπλές συσκευές
Υψηλής Ταχύτητας 25 – 500 Mb/s	**Συσκευές Video, Εξωτερικοί Δίσκοι**	Μεγάλο εύρος ζώνης Εγγυημένη καθυστέρηση Ευκολία χρήσης

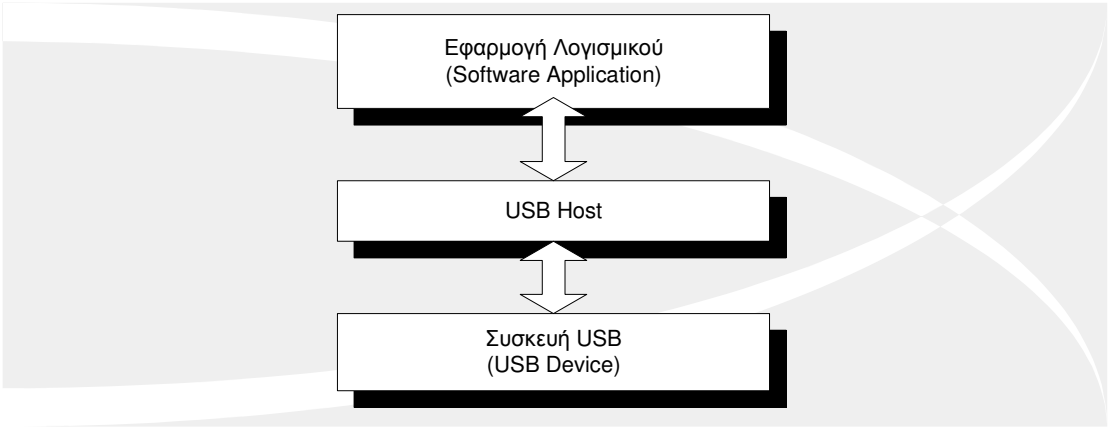
Παρακάτω παρουσιάζονται τα κυριότερα χαρακτηριστικά και πλεονεκτήματα του Universal Serial Bus 1.1:

- *Ευκολία προς τον χρήστη*:
  - Απόκρυψη των ηλεκτρικών λεπτομερειών του διαύλου από τον τελικό χρήστη.
  - Αυτόματη αναγνώριση, ρύθμιση και οδήγηση (μέσω κατάλληλου λογισμικού) των περιφερειακών συσκευών κατά τη σύνδεσή τους.
  - Δυναμική σύνδεση και αποσύνδεση των συσκευών από το σύστημα.
  - Υποστήριξη ενός μοναδικού μοντέλου καλωδιών και συνδέσεων.
- *Υποστήριξη μεγάλου εύρους συσκευών*:
  - Κάταλληλο για συσκευές με απαιτήσεις εύρους ζώνης από λίγα Kb/s μέχρι μερικά Mb/s.
  - Υποστήριξη ισόχρονων και ασύγχρονων τύπων μεταφοράς δεδομένων.
  - Υποστήριξη ταυτόχρονης λειτουργίας πολλαπλών συσκευών.
  - Υποστήριξη μέχρι και 127 συσκευών σε ένα δίαυλο.
  - Υποστήριξη πολλαπλών τύπων μεταφοράς δεδομένων.
  - Υποστήριξη Σύνθετων Συσκευών (*Compound Devices)*. Μια σύνθετη συσκευή αποτελείται από πολλές ανεξάρτητες συσκευές που υλοποιούνται ως μια φυσική συσκευή.
  - Ελαχιστοποίηση των Bits πρωτοκόλλου σε κάθε πακέτο USB μεγιστοποιώντας την χρησιμοποίηση του διαύλου.
- *Προτεραιότητα στις ισόχρονες μεταφορές δεδομένων*:
  - Εγγυημένο εύρος ζώνης και μικρή καθυστέρηση στις ισόχρονες μεταφορές δεδομένων. Κατάλληλο για συσκευές τηλεφωνίας, ήχου, κτλ.
  - Οι ισόχρονες μεταφορές δεδομένων μπορούν να χρησιμοποιούν μέχρι και το 90 % του εύρους ζώνης του διαύλου.

- *Ευελιξία:*
    - Υποστήριξη μεγάλου εύρους μεγεθών πακέτων δεδομένων, ανάλογα με το μέγεθος των καταχωρητών κάθε συσκευής.
    - Υποστήριξη μεγάλου εύρους ρυθμών μεταφοράς δεδομένων ανάλογα με τις ανάγκες κάθε συσκεύης.
- *Αξιοπιστία*:
    - Αυτόματος έλεγχος και χειρισμός των σφαλμάτων.
    - Αναγνώριση των ελαττωματικών συσκευών.
- Εμπορικότητα:
    - Ευκολία υλοποίησης του πρωτοκόλλου
    - Συμβατό με την αρχιτεκτονική Plug-and-Play
- Χαμηλό κόστος υλοποίησης:
    - Κατάλληλο για την υλοποίηση περιφερειακών χαμηλού κόστους.
    - Καλώδια και συνδέσεις χαμηλού κόστους.
    - Σχεδιασμένο για την μεγιστοποίηση της απόδοσης των περιφερειακών και του υλικού των PC.
- Αναβαθμισιμότητα:
    - Σχεδιασμένο έτσι ώστε ένα υπολογιστικό σύστημα να μπορεί να χρησιμοποιεί ταυτόχρονα πολλούς ελεγκτές USB.

1.2 Δομή του Συστήματος USB

Ένα Σύστημα USB αποτελείται από ένα Κεντρικό Υπολογιστικό Σύστημα, το οποίο αναφέρεται ως USB Host, τον Δίαυλο USB, και τις διάφορες Συσκευές USB *(USB Devices)* που είναι προσαρτημένες στον δίαυλο μεσώ Διακλαδωτών *(Hubs)*. Το USB επιτρέπει την ανταλλαγή δεδομένων μεταξύ του USB Host και των διαφόρων συσκευών. Όπως φαίνεται στο Σχήμα 1.2.1, οι διάφορες Εφαρμογές Λογισμικού (*Software Applications*) που εκτελούνται στο υπολογιστικό σύστημα, και οι οποίες απαιτούν μεταφορά δεδομένων προς/από τις συσκευές USB που είναι συνδεδεμένες στον δίαυλο, χρησιμοποιούν το λογισμικό και το υλικό του USB Host για την μετάφραση των αιτήσεων μεταφοράς δεδομένων σε κίνηση στον διάυλο συμβατή με το πρωτόκολλο USB και προς την επιθυμητή κατεύθυνση.

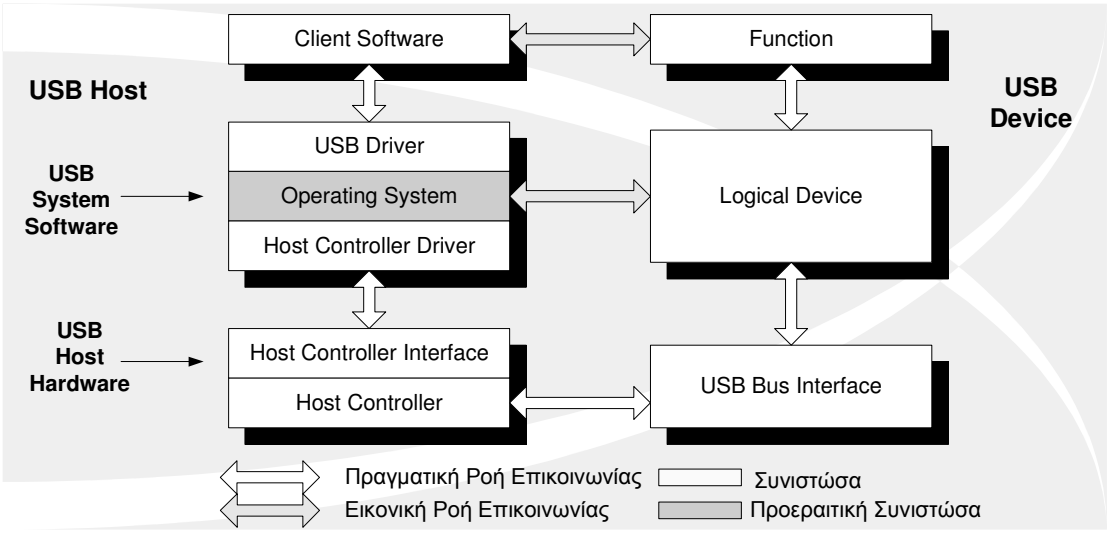


Σχήμα 1.2.1: Δομή του Συστήματος USB

Όπως θα περιγραφεί λεπτομερώς στα επόμενα Κεφάλαια του συγγράμματος, ο USB Host κατέχει ένα κεντρικό ρόλο σε κάθε Σύστημα USB. Ο USB Host καταλαμβάνει την κεντρική θέση στην φυσική τοπολογία ενός Συστήματος USB, παρέχοντας σημεία σύνδεσης των συσκευών σε αυτόν μέσω ενός Κεντρικού Διακλαδωτή *(Root Hub)*. Πέρα από αυτή την ιδιότητα, ο USB Host διαθέτει το κατάλληλο υλικό και λογισμικό για τη Διαχείριση *(Traffic Management)* και Χρονοδρομολόγηση *(Scheduling)* των Αιτήσεων Μεταφοράς Δεδομένων (*Data Transfer Requests*) προς ή από τις διάφορες συνδεδεμένες συσκευές. Οι αιτήσεις αυτές αναπαράγονται από κατάλληλο λογισμικό για κάθε συσκευή το οποίο είναι εγκατεστημένο στον USB Host και οδηγεί την κάθε συσκευή USB. Επιπλέον, ο USB Host αναλαμβάνει την Παρακολούθηση *(Monitoring)* και Ρύθμιση *(Configuration)* και διαχείριση *(Management)* των συκευών USB. Έτσι ο USB Host είναι σε κάθε χρονική στιγμή ενήμερος για το ποιές συσκευές είναι παρούσες στο σύστημα.

Όπως φαίνεται στο Σχήμα 1.2.2, η λογική δομή του USB Host είναι στρωματική. Κάθε Στρώμα *(Layer)*, το οποίο μπορεί να υλοποιείται με υλικό ή λογισμικό, εκτελεί διακριτές λειτουργίες και κατέχει έναν ιδιαίτερο ρόλο στο σύστημα. Η δομή του USB Host είναι αφαιρετική, δηλαδή κάθε στρώμα διαχειρίζεται τις συσκευές USB αγνοώντας τις λεπτομέρειες της υλοποίησης των κατώτερων στρωμάτων του USB Host. Με τον τρόπο αυτό, κάθε στρώμα του USB Host διατηρεί

μια αντίστοιχη αφαιρετική εικόνα των συσκευών που διαχειρίζεται και με την οποία επικοινωνεί εικονικά. Η πραγματική ροή επικοινωνίας γίνεται μέσω του διαύλου USB. Η Διαστρωματική Επικοινωνία *(Interlayer Communication)* επιτυγχάνεται μέσω Διαπροσωπειών *(Interfaces)* οι οποίες παρεμβάλλονται μεταξύ των γειτονικών στρωμάτων και παρέχουν κοινές δομές δεδομένων, μέσω των οποίων επικοινωνούν τα στρώματα ανταλλάσσοντας πληροφορίες. Σε ιεραρχική σειρά τα στρώματα και οι διαπροσωπείες ενός συστήματος USB είναι τα εξής:



Σχήμα 1.2.2: Στρωματική Δομή του USB Host και Ροή Επικοινωνίας με τη Συσκευή USB

- Λογισμικό Συσκευής *(Client Software)*. Το λογισμικό που οδηγεί μια συγκεκριμένη συσκευή που είναι συνδεδεμένη στον USB, παράγοντας τις αιτήσεις για μεταφορά δεδομένων προς και από τη συσκευή αυτή. Η εικόνα μιας συσκευής USB προς το στρώμα αντιστοιχεί στο σύνολο των Λειτουργιών (*Functions*) που υποστηρίζονται από τη συσκευή αυτή.

- Λογισμικό Οδήγησης του USB *(USB Driver, USBD)*: Το interface λογισμικού μεταξύ του USB System Software (βλέπε παρακάτω) και του Client Software. Παραλαμβάνει τις αιτήσεις για μεταφορά δεδομένων από τα διάφορα Client SW και τις παραδίδει στο USB System Software επιστρέφοντας το αποτέλεσμα (Status) της εκτέλεσής τους, όταν αυτή ολοκληρωθεί.

- Λογισμικό Συστήματος USB *(USB System Software)*. Το λογισμικό που είναι υπεύθυνο για την παρακολούθηση, ρύθμιση, και διαχείριση των συσκευών USB. Επιπλέον, το USB System Software εποπτεύει και ρυθμίζει τη συνολική λειτουργία του Συστήματος USB. Το USB System Software αποτελείται ως επί το πλείστον από το USBD και το HCD (βλέπε παρακάτω). Η εικόνα μιας συσκευής USB προς το στρώμα αυτό αντιστοιχεί στις λειτουργίες ελέγχου και ρύθμισης της συσκευής, χωρίς τις λεπτομέρειες της φυσικής διασύνδεσής της στο σύστημα. Η εικόνα αυτή της συσκευής αναφέρεται ως Λογική Συσκευή (*Logical Device*).

- Λογισμικό Οδήγησης του Ελεγκτή του Host *(Host Controller Driver, HCD)*: Η διαπροσωπεία λογισμικού μεταξύ του Host Controller του USB και του

USB System Software. Το HCD είναι υπεύθυνο γαι τον χειρισμό του Host Controller που οδηγεί, και την χρονοδρομολόγηση *(Scheduling)* των αιτήσεων μεταφοράς δεδομένων που πρόκειται να εκτελεστούν από τον HC. Η υλοποίηση του HCD εξαρτάται από την υλοποίηση του Host Controller.

- Ελεγκτής USB του Host *(USB Host Controller, HC)*. Ο Host Controller αποτελεί το υλικό του USB Host. Ο Host Controller εκτελεί το χρονοδιάγραμμα *(Schedule)* που του παραδίδει το λογισμικό και αναφέρει σε αυτό το αποτέλεσμα *(Status)* της εκτέλεσης κάθε αίτησης. Η ανταλλαγή δεδομένων μεταξύ του HCD και του HC πραγματοποιείται μέσω μιας δοαπροσωπείας υλικού-λογισμικού που διαθέτει κάθε Host Controller και η οποία αναφέρεται ως Host Controller Interface. Ο HC πραγματοποιεί την ροή δεδομένων προς/από τις συνδεδεμένες συσκευές, μέσω του διαύλου USB, επικοινωνώντας με την Διαπροσωπεία Διαύλου USB (*USB Bus Interface*) κάθε συσκευής, μέσω της οποίας πραγματοποιείται η φυσική σύνδεση της συσκευής USB στον δίαυλο. Τέλος, ο HC υλοποιεί το Root Hub του συστήματος USB.

1.3 Δομή και Περιεχόμενο της Εργασίας

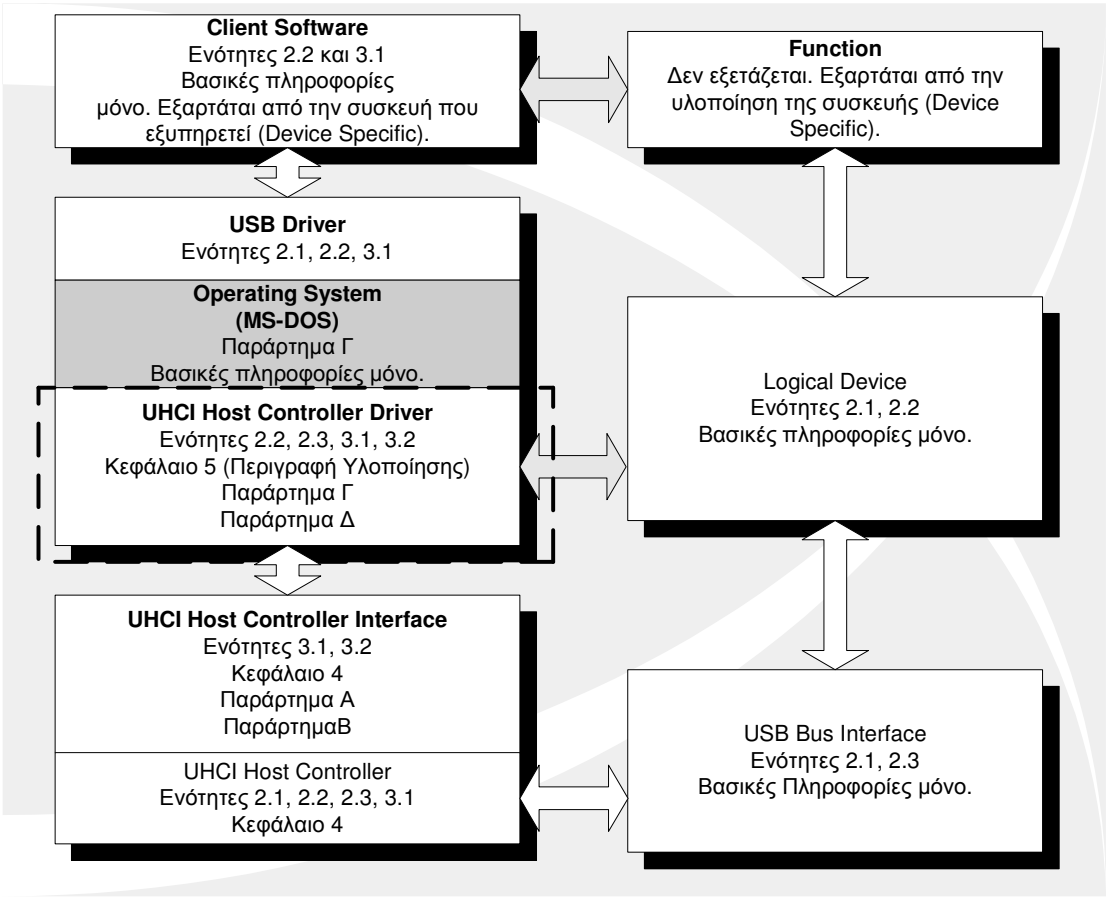
Στην παρούσα Διπλωματική Εργασία παρουσιάζονται αναλυτικά οι προδιαγραφές του Universal Serial Bus 1.1 που αφορούν την δομή και τη λειτουργία του USB Host, λόγω του ιδιαίτερου ρόλου του σε κάθε σύστημα USB. Η ανάλυση επικεντρώνεται στην περιγραφή των αρμοδιοτήτων των στρωμάτων του USB Host και του τρόπου με τον οποίο επιτυγχάνεται η διαστρωματική επικοινωνία. Ιδιαίτερη έμφαση δίνεται στις προδιαγραφές που αφορούν το Λογισμικό Οδήγησης του Ελεγκτή του Host (*Host Controller Driver*). Δεν μελετώνται οι προδιαγραφές του Universal Serial Bus 1.1 για τις συσκευές USB (*USB Device Framework*), αλλά δίνονται μόνο βασικές πληροφορίες που αφορούν την επικοινωνία τους με τον USB Host. Η εξέταση των ηλεκτρικών και μηχανικών χαρακτηριστικών του Universal Serial Bus είναι εκτός του θέματος της παρούσας Διπλωματικής Εργασίας. Στη συνέχεια, παρουσιάζονται οι προδιαγραφές του Universal Host Controller Interface *(UHCI)*, το οποίο είναι η πιο δημοφιλής υλοποίηση διαπροσωπείας υλικού-λογισμικού του USB Host Controller.

Με βάση τις προδιαγραφές του USB 1.1 και του UHCI αναπτύχθηκε ένα μέρος του Λογισμικού Οδήγησης του Ελεγκτή του Host (*HCD*) για το λειτουργικό σύστημα MS-DOS. Η υλοποίηση του λογισμικού επικεντρώθηκε στην χρονοδρομολόγηση των αιτήσεων μεταφορών δεδομένων και την κατάλληλη οδήγησή τους στον Host Controller. Ο κώδικας του λογισμικού μπορεί, με κάποιες μετατροπές, να χρησιμοποιηθεί για την υλοποίηση ενός HCD για άλλα λειτουργικά συστήματα (πχ Micro-Empix).

Στο Σχήμα 1.3.1 παρουσιάζεται η κατανομή της ύλης της Διπλωματικής Εργασίας σε σχέση με την δομή του Συστήματος USB. Στο σχήμα αυτό φαίνονται οι διάφορες συνιστώσες του συστήματος USB που πρέπει να μελετηθούν για την υλοποίηση ενός UHCI Host Controller Driver, με παραπομπές στα αντίστοιχα εδάφια του παρόντος συγγράμματος όπου μελετώνται. Σε διακεκομμένο πλαίσιο περικλείται το μέρος του λογισμικού του συστήματος USB που υλοποιήθηκε. Συνοπτικά το περιεχόμενο του παρόντος συγγράμματος είναι το εξής:

- Στο Κεφάλαιο 2 παρουσιάζονται οι προδιαγραφές του Universal Serial Bus 1.1. Στην Ενότητα 2.1 παρέχεται μια συνολική εικόνα του συστήματος USB, περιγράφοντας σε γενικά πλαίσια την φυσική τοπολογία, τον USB Host, τις συσκευές USB, τους διακλαδωτές USB, και τους τύπους μεταφοράς δεδομένων που υποστηρίζονται από το USB. Στην Ενότητα 2.2 περιγράφονται λεπτομερώς τα στρώματα του USB Host, η λογική τοπολογία και το μοντέλο ροής δεδομένων μεταξύ των οντοτήτων του συστήματος USB, και οι απαιτήσεις και προδιαγραφές των τύπων μεταφορών δεδομένων. Στην Ενότητα 2.3 παρουσιάζεται το Πρωτόκολλο Διαύλου του USB: Η μορφή των πακέτων USB, τα πεδία τους, ο τρόπος με τον οποίο μεταφέρονται τα δεδομένα, και ο συγχρονισμός μεταξύ Host-συσκευής. Οι παραπάνω προδιαγραφές και πληροφορίες θα αξιοποιηθούν για την υλοποίηση των δομών δεδομένων που αναπαριστούν την επικοινωνία μεταξύ του USB Host και των συσκευών και των αλγορίθμων που είναι απαραίτητοι για την χρονοδρομολόγηση των αιτήσεων μεταφοράς δεδομένων. Η παρουσίαση των

ηλεκτρικών και μηχανικών χαρακτηριστικών του USB είναι εκτός του θέματος της εργασίας.



Σχήμα 1.3.1: Κατανομή του Περιεχομένου της Διπλωματικής Εργασίας σε σχέση με τη Δομή του Συστήματος USB

- Στο Κεφάλαιο 3 παρουσιάζεται πιο αναλυτικά ο τρόπος επικοινωνίας μεταξύ του υλικού και του λογισμικού του USB Host. Στην Ενότητα 3.1 παρουσιάζονται με μεγαλύτερη λεπτομέρεια τα στρώματα του USB Host. Στην Ενότητα 3.2 παρουσιάζονται και συγκρίνονται τα γενικά χαρακτηριστικά των δύο διαθέσιμων εμπορικών υλοποιήσεων διαπροσωπειών υλικού-λογισμικού του USB 1.1: Universal Host Controller Interface (*UHCI*) και Open Host Controller Interface (*OpenHCI*). Όπως θα φανεί στην ενότητα αυτή, στην περίπτωση του προτύπου UHCI είναι απαραίτητη η διάπαση των αιτήσεων μεταφοράς δεδομένων σε μικρότερες αιτήσεις, έτσι ώστε κάθε αίτηση να αντιστοιχεί σε μια συναλλαγή (*Transaction*) στον δίαυλο. Για τον λόγο αυτό κρίθηκε απαραίτητη η μελέτη του Πρωτοκόλλου Διαύλου του USB στην Ενότητα 2.3 του προηγούμενου κεφαλαίου.

- Στο Κεφάλαιο 4 παρουσιάζονται λεπτομερώς οι προδιαγραφές του Universal Host Controller Interface. Στην Ενότητα 4.1 δίνεται μια συνολική εικόνα του τρόπου λειτουργίας του UHCI. Στην Ενότητα 4.2 παρουσίαζονται οι καταχωρητές του UHCI, μέσω των οποίων πραγματοποιείται η επικοινωνία μεταξύ υλικού και λογισμικού. Στην Ενότητα 4.3 παρουσιάζονται οι δομές

δεδομένων που υποστηρίζει το UHCI για την δόμηση του χρονοδιαγράμματος (*Schedule)* των αιτήσεων μεταφοράς δεδομένων που πρόκειται να εκτελέσει ο Host Controller. Στην Ενότητα 4.4 περιγράφονται οι αλγόριθμοι εκτέλεσης του χρονοδιαγράμματος από τον Host Controller. Στην Ενότητα 4.5 περιγράφονται οι Διακοπές (*Interrupts*) υλικού ενός UHCI Host Controller. Έτσι, στο κεφάλαιο αυτό συλλέγονται οι απαραίτητες πληροφορίες για τις δομές δεδομένων που χρησιμοποιούνται από το Host Controller Driver για την παράδοση του χρονοδιαγράμματος των αιτήσεων μεταφοράς δεδομένων στον UHCI Host Controller, καθώς και τον τρόπο με τον οποίο το λογισμικό μπορεί να διαχειριστεί τον Host Controller.

- Στο Κεφάλαιο 5 περιγράφεται το λογισμικό που συγγράφηκε για την οδήγηση του ενός UHCI Host Controller. Παρουσιάζονται αναλυτικά οι δομές δεδομένων και οι αλγόριθμοι που αναπτύχθηκαν, με διαγράμματα ροής, και λεπτομέρειες για τον τρόπο υλοποίησής τους. Ο κώδικας του λογισμικού περιέχεται στο συνοδευτικό CD-ROM της Διπλωματικής Εργασίας.

- Στο Κεφάλαιο 6 συνοψίζεται το περιεχόμενο της Διπλωματικής Εργασίας, αναφέρονται τα συμπεράσματα που προκύπτουν από αυτή, και προτείνονται μελλοντικές κατευθύνσεις για την υλοποίηση ενός συστήματος λογισμικού που να υποστηρίζει όλες τις λειτουργίες ενός Συστήματος USB 1.1.

- Στο Παράρτημα Α παρουσιάζονται ένα-προς-ένα τα Bits των καταχωρητών του UHCI.

- Στο Παράρτημα Β παρουσιάζονται ένα-προς-ένα τα Bits των δομών δεδομένων του UHCI.

- Στο Παράρτημα Γ δίνονται πληροφορίες για την διαχείριση συσκευών που είναι συνδεδεμένες στον δίαυλο PCI (*Peripheral Component Interconnect*). Αυτές οι πληροφορίες θα χρησιμοποιηθούν από το λογισμικό για την λήψη των παραμέτρων της επικοινωνίας με τον USB Host Controller.

Στο Παράρτημα Δ περιγράφεται το περιεχόμενο του συνοδευτικού CD-ROM της Διπλωματικής Εργασίας.

ΚΕΦΑΛΑΙΟ 2

Universal Serial Bus 1.1

Στο Κεφάλαιο αυτό περιγράφονται βασικοί κανόνες και αρχές του Universal Serial Bus 1.1. Η ανάλυση που ακολουθεί βασίζεται στο Universal Serial Bus Specification Revision 1.1 **[1]**, το οποίο περιγράφει τις προδιαγραφές του USB που τέθηκαν από τις εταιρίες Intel, Microsoft, Compaq και NEC. Η ανάλυση δίνει έμφαση σε θέματα που αφορούν τη μεταφορά δεδομένων μεταξύ των συσκευών του USB. Οι προδιαγραφές αυτές πρέπει να τηρούνται από το υλικό και το λογισμικό σε κάθε σύστημα USB. Έτσι, για την ανάπτυξη του λογισμικού της Διπλωματικής Εργασίας λήφθηκαν υπόψην οι προδιαγραφές του USB 1.1. Το Κεφάλαιο ξεκινάει με μια περιγραφή της συνολικής εικόνας και των χαρακτηριστικών του USB, συνεχίζει με την περιγραφή του μοντέλου ροής δεδομένων και των αρμοδιοτήτων των διαφόρων μερών του συστήματος USB καθώς και τύπων μεταφοράς δεδομένων που υποστηρίζονται από το USB 1.1, και ολοκληρώνεται με την περιγραφή των κανόνων του πρωτοκόλλου που διέπει τη μεταφορά δεδομένων στον δίαυλο *(Bus)* του USB. Δίνονται πληροφορίες μόνο για τα βασικά χαρακτηριστικά των συσκευών και διακλαδωτών *(Hubs)* του USB, και όχι για τον τρόπο χειρισμού τους. Δεν περιγράφονται τα ηλεκτρικά και μηχανικά χαρακτηριστικά του συστήματος. Για μια πλήρη περιγραφή του USB, ο αναγνώστης μπορεί να ανατρέξει στην σχετική βιβλιογραφία.

2.1 Συνολική Εικόνα του Συστήματος USB

Σε αυτή την ενότητα περιγράφονται συνοπτικά η αρχιτεκτονική και οι βασικές έννοιες του συστήματος USB. Το USB είναι ένας ενσύρματος δίαυλος που υποστηρίζει ανταλλαγή δεδομένων μεταξύ ενός Κεντρικού Υπολογιστικού Συστήματος *(Host)* και διαφόρων περιφερειακών συσκευών, οι οποίες είναι ταυτόχρονα προσπελάσιμες από το υπολογιστικό σύστημα. Τα συνδεδεμένα περιφερειακά μοιράζονται το εύρος ζώνης *(Bandwidth)* του USB μέσω ενός πρωτοκόλλου βασιζόμενο σε κουπόνια *(Tokens)*. Ο Host είναι υπεύθυνος για την χρονοδρομολόγηση *(Schedule)* της κίνησης των δεδομένων στον δίαυλο. Το USB επιτρέπει την σύνδεση, ρύθμιση *(Configuration)*, χρησιμοποίηση, και αποσύνδεση των περιφερειακών ενώ ο Host και άλλα περιφερειακά βρίσκονται σε λειτουργία.

2.1.1 Περιγραφή του Συστήματος USB

Ένα σύστημα USB περιγράφεται από τρία μέρη:

- Τη Διασύνδεση USB *(USB Interconnect)*
- Τις Συσκευές USB
- Τον USB Host

Το USB Interconnect είναι ο τρόπος με τον οποίο οι συσκευές συνδέονται και επικοινωνούν με τον Host. Περιλαμβάνει τα εξής:

- Τοπολογία του Bus *(Bus Topology)*: Το μοντέλο σύνδεσης μεταξύ των συσκευών και του Host.

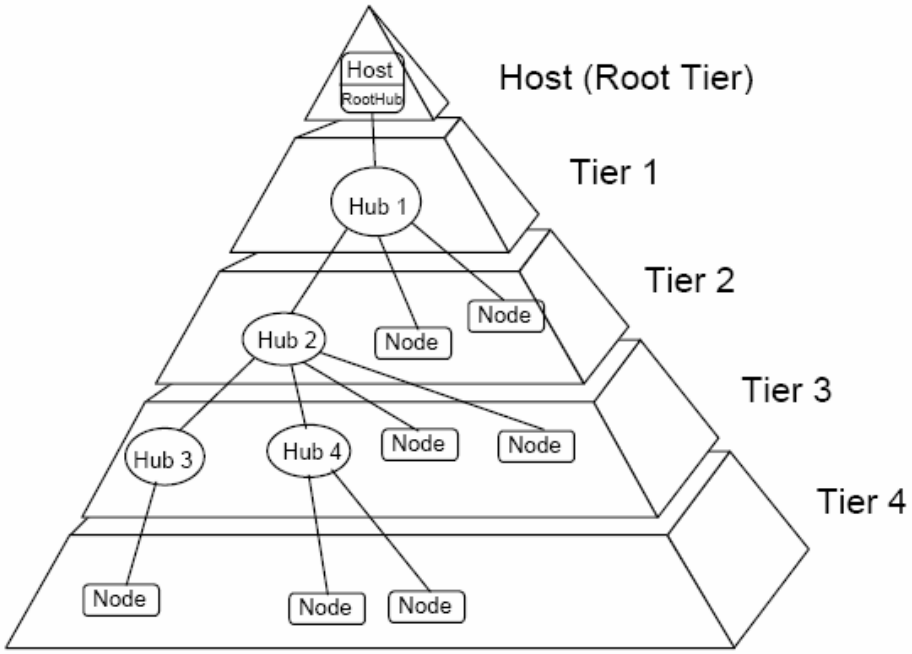
- Διαστρωματικές σχέσεις του USB *(Inter-layer Relationships)*: Οι επεξεργασίες που εκτελούνται σε κάθε στρώμα του USB.

- Μοντέλο ροής δεδομένων (*Data Flow Model*): Ο τρόπος με τον οποίο μεταφέρονται τα δεδομένα στο σύστημα USB.

- Χρονοδρομολόγηση του USB: Το USB παρέχει μια κοινή διασύνδεση για όλες τις συσκευές. Η πρόσβαση στο USB χρονοδρομολογείται για την υποστήριξη μεταφοράς δεδομένων με σταθερό ρυθμό μετάδοσης, και την κατάλληλη εξυπηρέτηση των αιτήσεων μεταφοράς δεδομένων.

2.1.2 Τοπολογία του Διαύλου USB

Το USB συνδέει τις συσκευές με τον Host. Η τοπολογία του bus έχει τη μορφή βαθμίδων αστέρα *(Tiered Star Topology)*, όπως φαίνεται στο Σχήμα 2.1.1. Στο κέντρο κάθε αστέρα βρίσκεται ένα Hub. Κάθε συσκευή συνδέεται στον Host είτε άμεσα, είτε μέσω ενός ή περισσότερων Hub.



Σχήμα 2.1.1: Τοπολογία του Διαύλου USB

2.1.2.1 Ο Host του USB

Σε κάθε σύστημα USB υπάρχει πάντα μόνο ένας Host, ο οποίος αποτελείται από υλικό και λογισμικό. Το hardware του Host, μέσω του οποίου ο Host συνδέεται στον δίαυλο, ονομάζεται Host Controller *(HC)*. Ένα κεντρικό hub *(Root Hub)* είναι ενσωματωμένο στον HC για την παροχή ενός ή περισσότερων σημείων σύνδεσης στον Host.

2.1.2.2 Συσκευές USB

Οι συσκεύες USB μπορεί να είναι:

- Διακλαδωτές *(Hubs)*, οι οποίοι παρέχουν πρόσθετα σημεία σύνδεσης στο USB.

- Συσκευές που λειτουργούν ως δέκτες ή πομποί δεδομένων στον δίαυλο *(Functions)*, όπως εκτυπωτές, modems, πληκτρολόγια κτλ.

Οι συσκευές πρέπει να είναι συμβατές με το πρωτόκολλο του USB, να αντιδρούν σε απαραίτητες λειτουργίες του USB, όπως ρύθμιση και επανεκκίνηση, και να παρέχουν στο USB πληροφορίες που να περιγράφουν τα χαρακτηριστικά τους. Ο δίαυλος USB υποστηρίζει σηματοδοσία πλήρης ταχύτητας *(full-speed)* στα 12 Mb/s και σηματοδοσία χαμηλής ταχύτητας *(low-speed)* στα 1.5 Mb/s. Έτσι οι συσκευές διακρίνονται σε:

- Συσκευές Πλήρους ταχύτητας *(Full-speed Devices)*. Υποστηρίζουν ρυθμούς μετάδοσης μέχρι 12 Mb/s και υπόκεινται σε σηματοδοσία full-speed. Τα hubs είναι full-speed συσκευές.

- Συσκευές Χαμηλής Ταχύτητας *(Low-speed Devices)*. Υποστηρίζουν ρυθμούς μετάδοσης μέχρι 1.5 Mb/s και υπόκεινται σε σηματοδοσία low-speed.

2.1.3 Πρωτόκολλο του Διαύλου USB

Η κατεύθυνση μιας μεταφοράς δεδομένων μπορεί να είναι από τον Host προς μια συσκευή, ή από μια συσκευή προς τον Host. Όλες οι μεταφορές δεδομένων ξεκινάνε με την αποστολή ενός κουπονιού *(token)* από τον Host, με βάση το Schedule. Το token περιγράφει το είδος και την κατεύθυνση της μεταφοράς δεδομένων (από τον host προς τη συσκευή ή από τη συσκευή προς τον host), και τη διεύθυνση της συσκευής και τον αριθμό του Τερματικού Σημείου *(Endpoint)* της συσκευής που ανταλλάσει δεδομένα με τον Host. Κάθε συσκευή μπορεί να υποστηρίζει πολλαπλά τερματικά σημεία, τα οποία αποτελούν τις θύρες επικοινωνίας της συσκευής με τον Host μέσω του διάυλου USB. Όλες οι συσκευές λαμβάνουν το πακέτο token και αποκωδικοποιούν το πεδίο διεύθυνσης. Η συσκευή που το πεδίο διεύθυνσης του token συμπίπτει με τη διεύθυνσή της θα ετοιμαστεί για μεταφορά δεδομένων. Ο πομπός και ο δέκτης υποδεικνύονται από το token. Ο πομπός στέλνει ένα πακέτο δεδομένων *(Data)*, ή υποδεικνύει ότι δεν έχει δεδομένα για αποστολή. Ο δέκτης λαμβάνει τα δεδομένα, αν υπάρχουν, και στέλνει ένα πακέτο χειραψίας *(Handshake)* για να δηλώσει στον πομπό εάν η μεταφορά δεδομένων ήταν επιτυχής ή όχι. Η παραπάνω διαδικασία ονομάζεται Transaction *(συναλλαγή)* και περιλαμβάνει ανταλλαγή τριών πακέτων το πολύ (Token, Data, Handshake). Όπως θα περιγραφτεί στην ενότητα 2.3, ένα transaction μπορεί να ολοκληρωθεί με ανταλλαγή λιγότερων από τρία πακέτα.

Το μοντέλο μεταφοράς δεδομένων του USB μεταξύ του host και του endpoint μιας συσκευής αναφέρεται ως «Σωλήνωση» *(Pipe)*. Υπάρχουν δύο είδη pipes: Message *(Μηνυμάτων)* και Stream *(Ροής Δεδομένων)*. Τα stream δεδομένα δεν έχουν κάποια συγκεκριμένη δομή, σε αντίθεση με τα message δεδομένα. Επιπλέον, τα pipes εξαρτώνται από το εύρος ζώνης, τον τύπο της μεταφοράς δεδομένων, και από διάφορα χαρακτηριστικά των endpoints, όπως η κατεύθυνση μεταφοράς δεδομένων που υποστηρίζουν και το μέγεθος των καταχωρητών (*Buffers*) του. Τα περισσότερα pipes δημιουργούνται όταν η συσκευή ρυθμιστεί. Ένα message pipe, το Default Control Pipe *(Πρωτεύον Pipe Ελέγχου)*, πάντα δημιουργείται μόλις μια συσκευή συνδεθεί στο USB, έτσι ώστε να υπάρχει πρόσβαση στις πληροφορίες ρύθμισης *(Configuration)*, ελέγχου *(Control)*, και κατάστασης *(Status)* της συσκευής.

2.1.4 Αξιοπιστία του USB

Το USB έχει τις εξής ιδιότητες που το καθιστούν αξιόπιστο:

- Προστασία με πεδία ανίχνευσης σφαλμάτων που εφαρμόζονται στα πεδία ελέγχου και δεδομένων των πακέτων USB, τα οποία παρέχουν 100% προστασία από σφάλματα ενός ή δύο bit.

- Ανίχνευση της σύνδεσης και αποσύνδεσης συσκευών, και αυτόματη ρύθμισή τους.

- Αποκατάσταση σε περίπτωση χαμένων ή λανθασμένων πακέτων USB. Το hardware επαναλαμβάνει ένα αποτυχημένο transaction τρεις φορές το πολύ, πριν ενημερώσει το λογισμικό της συσκεύης στον Host *(Client Software)* για να διαχειριστεί εκείνο το σφάλμα. Ο τρόπος με τον οποίο το λογισμικό θα διαχειριστεί το σφάλμα εξαρτάται από την υλοποίηση.

- Συγχρονισμός μεταξύ πομπού και δέκτη για τη λήψη των πακέτων δεδομένων στη σωστή σειρά.

- Έλεγχος της ροής των δεδομένων για την εξασφάλιση ισόχρονων μεταδόσεων και διαχείριση των buffers.

- Παρακολούθηση της κατάστασης των συνδεδεμένων συσκευών και κατάλληλη διαχείρισή τους.

2.1.5 Ρύθμιση του Συστήματος

Το USB υποστηρίζει τη σύνδεση και αποσύνδεση των συσκευών οποιαδήποτε στιγμή. Επομένως το λογισμικό του συστήματος πρέπει να είναι ικανό να διαχειρίζεται δυναμικά τις αλλαγές στην τοπολογία του bus.

Όλες οι USB συσκευές συνδέονται στις θύρες *(Ports)* των hubs. Τα hubs ανιχνεύουν την αλλαγή της κατάστασης σύνδεσης των θυρών τους *(Port Connection Status)*. Ο host διαβάζει την κατάσταση σύνδεσης των θυρών. Στην περίπτωση

ανίχνευσης σύνδεσης, ο host ενεργοποιεί *(Enable)* τη θύρα, αναθέτει στη συσκευή μια κανονική διεύθυνση και διαπιστώνει αν η συσκευή είναι hub ή function. Το control pipe της συσκευής θα χρησιμοποιεί τη διεύθυνση και το endpoint 0 της συσκευής. Αν η συσκευή είναι hub και υπάρχουν συσκευές συνδεδεμένες στις θύρες του, ακολουθείται η παραπάνω διαδικασία για κάθε συσκευή του hub. Αν η συσκευή είναι function, το λογισμικό του host θα εκτελέσει τις περεταίρω απαραίτητες ρυθμίσεις για τη συσκευή.

Όταν μια συσκευή αποσυνδέεται από μια θύρα ενός hub, το hub απενεργοποιεί *(Disable)* τη θύρα και ενημερώνει τον host, ώστε να χειριστεί την αποσύνδεση της συσκευής μέσω του λογισμικού του συστήματος USB. Αν η αποσυνδεόμενη συσκευή είναι hub, το λογισμικό χειρίζεται την αποσύνδεση του hub και όλων των συσκευών που συνδέονταν στον host μέσω του hub.

Η διαδικασία ανάθεσης διευθύνσεων στις συσκευές και ανίχνευσης σύνδεσης ή αποσύνδεσης στον δίαυλο ονομάζεται Bus Enumeration *(Απαρίθμηση Διαύλου)*.

2.1.6 Τύποι Μεταφοράς Δεδομένων

Το USB υποστηρίζει μεταφορά δεδομένων και πληροφοριών ελέγχου χρησιμοποιώντας pipes μονής ή διπλής κατεύθυνσης. Η μεταφορά δεδομένων λαμβάνει μέρος μεταξύ του λογισμικού του host και ενός συγκεκριμένου endpoint της συσκευής. Γενικά, η ροή δεδομένων σε ένα pipe δεν εξαρτάται από τη ροή δεδομένων σε άλλα pipes. Μια συσκευή μπορεί να έχει πολλά pipes.

Το USB διακρίνει τέσσερα είδη μεταφοράς δεδομένων *(Transfer Types)*:

- Control Transfers *(Μεταφορές Ελέγχου)*: Χρησιμοποιούνται για τη ρύθμιση μιας συσκευής κατά τη σύνδεσή της και μπορούν να χρησιμοποιηθούν για άλλους σκοπούς, όπως για τον έλεγχο των άλλων pipes της συσκευής.

- Bulk Transfers *(Ογκώδεις Μεταφορές)*: Χρησιμοποιούνται για τη μεταφορά μεγάλων ποσοτήτων δεδομένων για τα οποία δεν είναι κρίσιμο να μεταφερθούν γρήγορα. Χρησιμοποιούνται για συσκευές όπως εκτυπωτές, scanners κτλ.

- Interrupt Transfers *(Μεταφορές Διακοπής)*: Χρησιμοποιούνται για τη μεταφορά μικρών ποσοτήτων δεδομένων για τα οποία η καθυστέρηση της μεταφοράς τους είναι κρίσιμος παράγοντας. Χρησιμοποιούνται για συσκευές που παρέχουν χαρακτήρες ή συντεταγμένες, όπως πληκτρολόγια ή συσκευές δεικτοδότησης *(Pointing Devices)*.

- Isochronous Transfers *(Ισόχρονες Μεταφορές)*: Χρησιμοποιούνται για συσκευές που απαιτούν μεταφορά δεδομένων σε πραγματικό χρόνο *(Real-time)*, όπως μικρόφωνα ή κάμερες Web. Χαρακτηρίζονται από σταθερό ρυθμό μεταφοράς δεδομένων. Δεσμεύουν ένα προκαθορισμένο και σταθερό bandwidth με προκαθορισμένη καθυστέρηση.

Οι τύποι μεταφοράς Control, Bulk και Interrupt απαιτούν ανίχνευση και διόρθωση λαθών στα δεδομένα που μεταφέρονται. Οι Isochronous Transfers δεν υποστηρίζουν ανίχνευση λαθών, καθώς ο κρίσιμος παράγοντας για αυτά είναι η μεταφορά δεδομένων σε πραγματικό χρόνο. Ένα pipe μπορεί να υποστηρίζει μόνο έναν από τους παραπάνω τύπους μεταφοράς δεδομένων.

2.1.7 Περιγραφή των Συσκευών USB

Η προσπέλαση μιας συσκευής USB γίνεται μέσω της διεύθυνσής της, που της ανατίθεται όταν μια συσκευή συνδέται και απορριθμείται (*Enumerated*). Κάθε συσκευή πρέπει να υποστηρίζει ένα control pipe στο endpoint 0. Όλες οι συσκευές USB υποστηρίζουν τον ίδιο μηχανισμό πρόσβασης για την προσπέλαση πληροφοριών σε αυτό το control pipe. Μέσω αυτού του pipe διαβάζονται από τον host οι απαραίτητες πληροφορίες για την πλήρη περιγραφή της συσκευής. Οι πληροφορίες αυτές χωρίζονται στις παρακάτω κατηγορίες:

- Standard *(Καθιερωμένες)*: Αυτές οι πληροφορίες, οι οποίες περιέχονται σε κάθε συσκευή USB, περιγράφουν τον κατασκευαστή της συσκευής *(Vendor Identification)*, την κλάση *(Class)* της συσκευής, και τον τρόπο τροφοδοσίας (αυτόνομη ή τροφοδοσία από το bus) της συσκευής *(Power Management)*. Οι περιγραφές των Device *(συσκευή)*, Configuration *(ρύθμιση)*, Interface *(διαπροσωπεία)*, και Endpoint (*Τερματικά Σημεία)* περιέχουν πληροφορίες για τη ρύθμιση της συσκευής.

- Class *(Κλάση)*: Οι πληροφορίες που περιέχονται εξαρτώνται από το Class της συσκευής.

- USB Vendor *(Κατασκευαστής)*: Ο κατασκευαστής της συσκευής μπορεί να βάλει εδώ ό,τι πληροφορίες επιθυμεί.

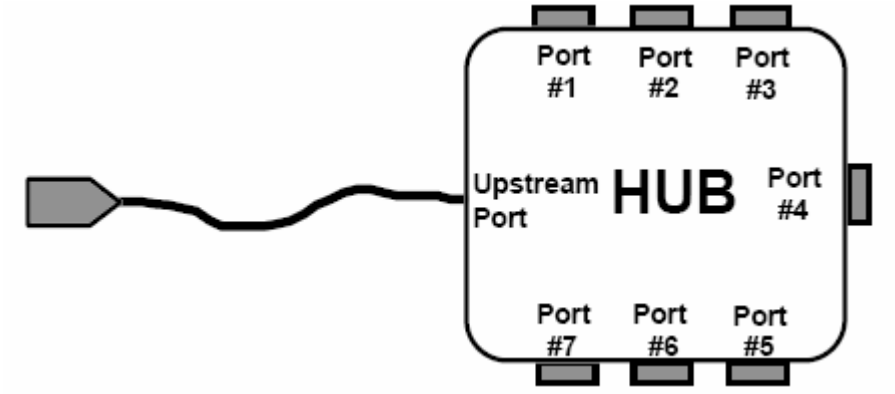
Επιπλέον, κάθε συσκευή περιέχει πληροφορίες ελέγχου *(control)* και κατάστασης *(status)*.

2.1.8 Διακλαδωτές του USB

Οι διακλαδωτές *(Hubs)* είναι μια ειδική κατηγορία συσκευών USB που μετατρέπουν ένα σημείο σύνδεσης σε πολλά, αυξάνοντας έτσι τον αριθμό των συσκευών στον δίαυλο. Στο Σχήμα 2.1.2 παρουσιάζεται ένα τυπικό hub. Τα σημεία σύνδεσης ονομάζονται θύρες *(Ports)*.

Η θύρα που συνδέει το hub με τον host ονομάζεται Upstream Port. Οι άλλες θύρες, στις οποίες συνδέονται οι συσκευές αναφέρονται ως Downstream Ports. Αντίστοιχα η κίνηση δεδομένων από το hub προς τον host αναφέρεται Upstream κίνηση, σε αντίθεση με την Downstream κίνηση. Τα hubs επιτρέπουν την σύνδεση και αποσύνδεση συσκευών στα Downstream Ports οποιαδήποτε στιγμή, και παρέχουν ηλεκτρική ισχύ στις συνδεδεμένες συσκευές. Σε ένα Downstream Port

μπορεί να συνδέεται μια full-speed ή low-speed συσκευή. Τα hubs απομονώνουν τις low-speed συσκευές από την full-speed σηματοδοσία στον δίαυλο.



Σχήμα 2.1.2: Μορφή ενός Τυπικού Hub

Τα hubs αποτελούνται από δύο μέρη: τον *Ελεγκτή του Hub (*Hub Controller*)* και τον *Επαναλήπτη του Hub (*Hub Repeater*)*. Ο Hub Repeater είναι ένα switch ανάμεσα στο upstream port και τα downstream ports. Ο Hub controller περιέχει τους καταχωρητές για την επικοινωνία του hub με τον host. Διάφορες εντολές, που εξαρτώνται από την υλοποίηση του hub, επιτρέπουν στον host να ρυθμίζει το hub, και να ελέγχει ή να διαβάσει την κατάσταση των θυρών του.

2.1.9 Χρονικά Πλαίσια (Frames) του USB

Σε ένα σύστημα USB ο πραγματικός χρόνος διαρείται σε πλαίσια *(Frames)*, μέσα στα οποία γίνονται μεταφορές δεδομένων στον δίαυλο. Στο τέλος ενός frame ο δίαυλος πρέπει να είναι σε ανενεργή *(Idle)* κατάσταση, δηλαδή να μην πραγματοποιείται μεταφορά δεδομένων.

Η χρονική διάρκεια ενός frame είναι 1 ms. Ο Host Controller μπορεί να τροποποιήσει σε μικρό βαθμό την τιμή αυτή (1.0 ms ± 0.0005 ms) εάν αυτό κριθεί απαραίτητο. Τα όρια ενός frame ορίζονται αυστηρά από το USB. Η αρχή ενός πλαισίου καθορίζεται με την αποστολή από τον HC ενός ειδικού πακέτου, του πακέτου SOF *(Start-of-frame, Αρχή-του-πλαισίου)*. Ο HC αριθμεί τα frames και ενσωματώνει τον αριθμό κάθε frame (*Frame Number)* στο αντίστοιχο πακέτο SOF. Οι συσκευές λαμβάνουν το πακέτο SOF και το αποκωδικοποιούν. Οι συσκευές των οποίων η λειτουργία είναι ανεξάρτητη από το frame number αγνοούν το πακέτο και δεν το αποκωδικοποιούν. Το τέλος του frame *(EOF, End-of-frame)* συμβαίνει όταν σταλθεί το πακέτο SOF του επόμενου frame.

Τα Hubs παρακολουθούν την κίνηση των δεδομένων στον δίαυλο. Όλα τα hubs περιέχουν ένα εσωτερικό μετρητή ο οποίος μετράει τον χρόνο μέσα σε ένα frame και μηδενίζεται όποτε λαμβάνεται ένα πακέτο SOF. Ο μετρήτης αυτός παράγει το σημείο EOF2 το οποίο εντοπίζεται πολύ κοντά στο τέλος του frame. Η ακριβής θέση του σημείου EOF2 είναι εκτός του θέματος του παρόντος συγγράμματος. Εάν στο σημείο EOF2 ένα hub εντοπίσει κίνηση σε μια θύρα του, τότε απενεργοποιεί τη θύρα αυτή,

εμποδίζοντας τη μεταφορά δεδομένων, και αναφέρει ένα σφάλμα «φλυαρίας» *(Babble Error)* στον Host.

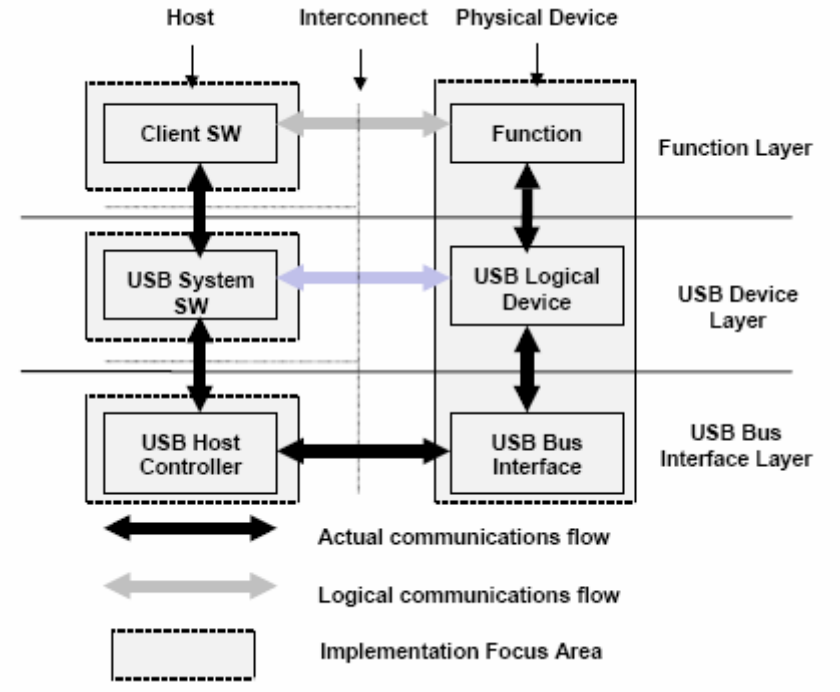
Το λογισμικό που οδηγεί τον HC είναι υπεύθυνο για την δρομολόγηση των αιτήσεων για μεταφορά σε ένα frame, με τέτοιο τρόπο έτσι ώστε αυτά να μην ξεπερνούν τα όριά του. Επιπλέον, ο HC σε ειδικές περιπτώσεις μπορεί να ελέγχει εάν ένα πακέτο προς αποστολή μπορεί να χωρέσει στο τρέχον frame και να πράττει ανάλογα. Αυτό το θέμα περιγράφεται λεπτομερώς στο Κεφάλαιο 4.

2.2 Μοντέλο Ροής Δεδομένων

Στην παρούσα ενότητα πραγματοποιείται μια αναλυτική παρουσίαση της λογικής δομής του USB Host, η οποία περιγράφτηκε συνοπτικά η δομή του USB Host, δίνοντας έμφαση στον τρόπο με τον οποίο κάθε συνιστώσα του λογισμικού του συστήματος USB διαχειρίζεται τη ροή επικοινωνίας μεταξύ του USB Host και μιας συσκευής-στόχου (*Target Device*). Η ανάλυση χρησιμοποιεί την ακριβή ορολογία που χρησιμοποιείται στην σχετική βιβλιογραφία **[1]**. Περιγράφονται οι Σωληνώσεις (*Pipes*) του συστήματος USB, που αποτελούν τις δομές δεδομένων που διατηρεί το λογισμικό του USB Host, οι οποίες περιέχουν όλες τις πληροφορίες που περιγράφουν τα χαρακτηριστικά της ροής δεδομένων μεταξύ του USB Host και των Endpoints των συσκευών που είναι συνδεδεμένες στο USB. Η ενότητα ολοκληρώνεται με την παρουσίαση των πεδιών εφαρμογών και των ιδιοτήτων των τύπων μεταφοράς δεδομένων που υποστηρίζονται από το USB 1.1. Επίσης παρουσιάζονται οι απαιτήσεις ό,τι αφορά την εξυπηρέτηση των αιτήσεων μεταφοράς δεδομένων κάθε τύπου μεταφοράς, οι οποίες θα αξιοποιηθούν κατά την υλοποίηση του λογισμικού της Διπλωματικής Εργασίας για την χρονοδρομολόγηση των αιτήσεων μεταφοράς δεδομένων. Δεν αναλύεται το πρωτόκολλο διαύλου του USB (*USB Protocol*), δηλαδή οι κανόνες βάσει των οποίων κινούνται τα δεδομένα στο USB. Το πρωτόκολλο USB παρουσιάζεται στην επόμενη ενότητα.

2.2.1 Στρωματική Δομή του Συστήματος USB

Όπως φαίνεται στο Σχήμα 2.2.1, το σύστημα USB αποτελείται στρώματα *(Layers)*, καθένα από το οποία εκτελεί διαφορετικές λειτουργίες.



Σχήμα 2.2.1: Στρωματική Δομή του Συστήματος USB

Τα βασικά μέρη *(Focus Areas)* από τα οποία αποτελείται ένα σύστημα USB είναι:

- Φυσική Συσκευή USB *(USB Physical Device)*: Υλικό στο τέλος ενός καλωδίου USB που υλοποιεί μια λειτουργία *(Function)* χρήσιμη προς τον χρήστη.

- Λογισμικό Συσκευής *(Client Software)*: Λογισμικό που εκτελείται στον Host και οδηγεί την USB συσκευή.

- Λογισμικό Συστήματος USB *(USB System Software)*: Λογισμικό που υποστηρίζει το USB σε ένα Λειτουργικό Σύστημα.

- Ελεγκτής USB του Host *(USB Host Controller, HC)*: Το υλικό και το λογισμικό που επιτρέπει στις συσκευές USB να συνδεθούν στον Host.

Σύμφωνα με το Σχήμα 2.2.1, η επικοινωνία του Host με μία συσκευή απαιτεί αλληλεπίδραση μεταξύ των στρωμάτων του συστήματος USB. Τα στρώματα αυτά είναι:

- Στρώμα Διαπροσωπείας του Διαύλου USB *(USB Bus Interface Layer)*: Παρέχει τη φυσική σύνδεση του Host με την USB συσκευή.

- Στρώμα Συσκευής *(USB Device Layer)*: Είναι η εικόνα που έχει το USB System Software για γενικές λειτουργίες πάνω στη συσκευή.

- Στρώμα Λειτουργίας *(Function Layer)*: Παρέχει πρόσθετες δυνατότητες στον Host μέσω του Client Software.

Τα δύο ανώτερα στρώματα του USB έχουν μια δική τους εικόνα της Λογικής Επικοινωνίας *(Logical Communication)* στο στρώμα τους και χρησιμοποιούν το USB Bus Interface Layer για την πραγματική μεταφορά δεδομένων.

2.2.2 Λογική Τοπολογία του USB

Στο Σχήμα 2.2.1 παρουσιάζεται η λογική σύνθεση του Host και των συσκευών του USB. Σε κάθε σύστημα USB υπάρχει μόνο ένας Host, ο οποίος αποτελεί το κεντρικό σημείο του συστήματος. Όπως περιγράφτηκε προηγουμένως, ο Host του USB αποτελείται από τα εξής λογικά μέρη (από κάτω προς τα πάνω):

- Host Controller

- USB System Software

- Client Software

Κάθε συσκευή αποκτά πρόσβαση στον δίαυλο, μόνο αν αυτό επιτραπεί από τον Host. Επίσης, ο Host είναι υπεύθυνος για την παρακολούθηση των αλλαγών στην φυσική τοπολογία του USB και τον έλεγχο των USB συσκευών.

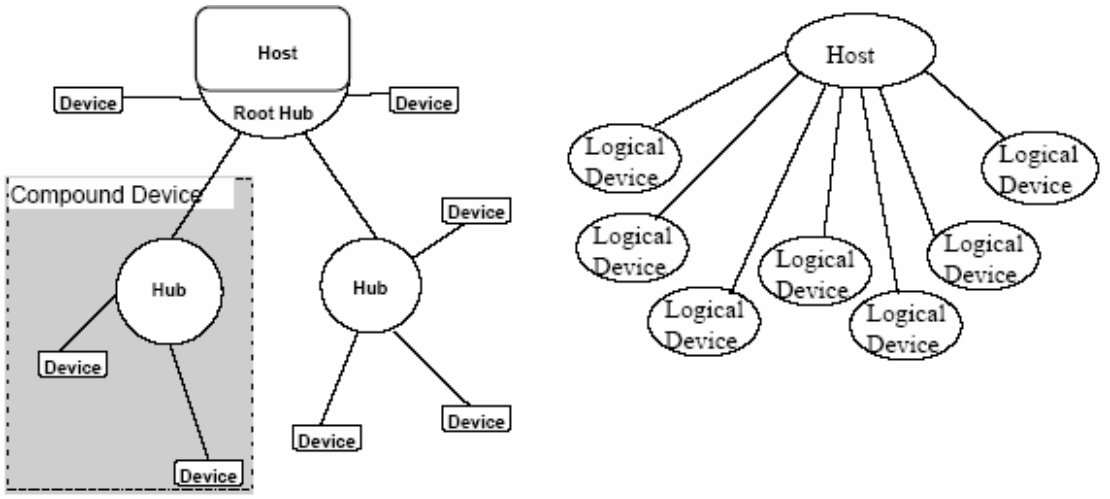
Μία φυσική συσκευή αποτελείται από τα εξής λογικά μέρη (από κάτω προς τα πάνω, βλέπε Σχήμα 2.2.1):

- Διαπροσωπεία του Διαύλου USB *(USB Bus Interface*): Παρέχει τη φυσική σύνδεση της συσκευής με τον host μέσω του USB.

- Λογική Συσκευή USB *(USB Logical Device)*: Η εικόνα της συσκευής στον host, χωρίς τις λεπτομέρειες της φυσικής διασύνδεσης στο USB, για τον έλεγχο και τη ρύθμιση της συσκευής.

- Λειτουργία *(Function)*: Ένα σύνολο από interfaces που χειρίζεται ο host ώστε να εκτελέστουν οι λειτουργίες της συσκευής.

Οι λειτουργίες που μπορούν να παρέχουν οι φυσικές συσκευές είναι πολλές. Όμως όλες οι λογικές συσκευές ελέγχονται από τον Host με τον ίδιο τρόπο, παρουσιάζοντας στον Host το ίδιο interface. Ενώ η φυσική τοπολογία ενός συστήματος USB, που παρουσιάστηκε στην ενότητα 2.1, έχει τη μορφή βαθμίδων αστέρα, ο Host επικοινωνεί με κάθε λογική συσκευή σαν να ήταν άμεσα συνδεδεμένη σε ένα port του root hub. Στο παρακάτω σχήμα παρουσιάζεται ένα παράδειγμα όπου φαίνεται η φυσική τοπολογία ενός συστήματος USB και η αντίστοιχη λογική τοπολογία.

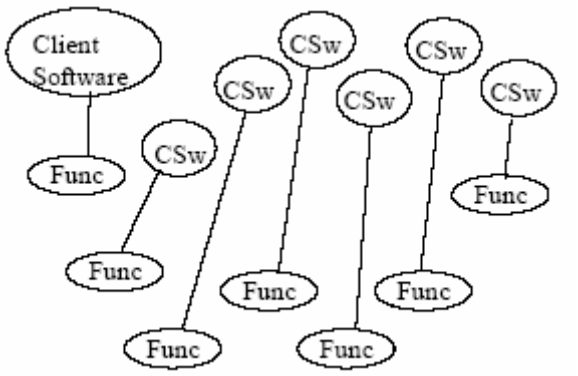


Σχήμα 2.2.2: Φυσική και Λογική Τοπολογία ενός Συστήματος USB

Στο παραπάνω παράδειγμα φαίνεται και μια ειδική κατηγορία φυσικών συσκευών, οι Σύνθετες Συσκευές *(Compound Devices)*. Οι συσκευές αυτές περιέχουν ένα εσωτερικό hub, στο οποίο συνδέονται μόνιμα πολλά functions. Για τον host, οι σύνθετες συσκευές είναι ισοδύναμες με ένα κοινό hub στο οποίο συνδέονται τα functions.

Τέλος το Client Software ενός function επικοινωνεί με το function σαν να υπάρχει μόνο αυτό στον δίαυλο, όπως στο Σχήμα 2.2.3, το οποίο αντιστοιχεί στο σύστημα USB του Σχήματος 2.2.2. Κατά τη λειτουργία του δεν επηρρεάζεται από άλλες συσκευές που μπορεί να συνδέονται στο USB.

Έτσι ο κατασκευαστής της συσκευής που συγγράφει το Client Software δεν χρειάζεται να ασχοληθεί με τον τρόπο με τον οποίο γίνεται η ροή δεδομένων μεταξύ του host και της συσκευής, αλλά μόνο με θέματα που αφορούν τον έλεγχο της συσκευής.



Σχήμα 2.2.3: Εικονική Επικοινωνία μεταξύ του Client Software και του αντίστοιχου Function

Από τα παραπάνω φαίνεται η αφαιρετική *(abstract)* δομή του USB. Ενώ όλες οι πραγματικές μεταφορές δεδομένων γίνονται μέσω των κατώτερων στρωμάτων του συστήματος USB, οι διάφορες οντότητες επικοινωνούν με οντότητες του ίδιου στρώματος χωρίς να ενδιαφέρονται για τις λεπτομέρειες των κατώτερων στρωμάτων.

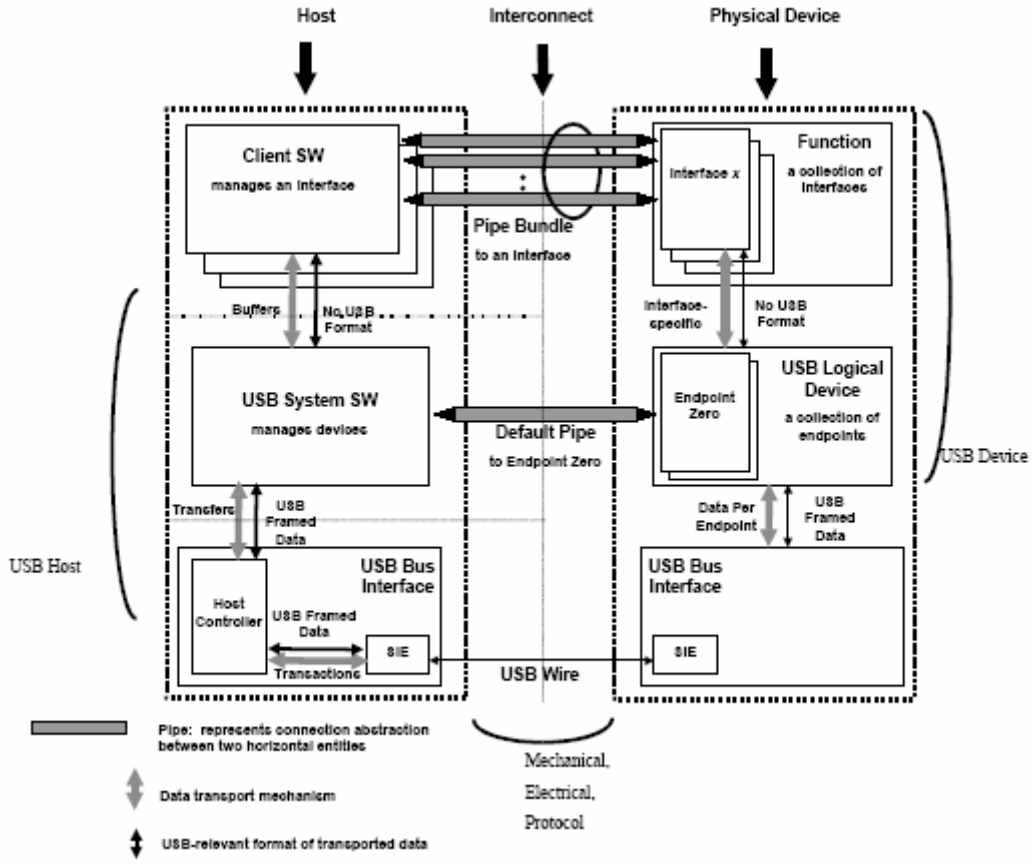
2.2.3 Ροή Επικοινωνίας στο USB

Το USB παρέχει επικοινωνία μεταξύ του λογισμικού της συσκευής στον host και του function που οδηγεί. Τα functions μπορεί να έχουν διαφορετικές απαιτήσεις ροής επικοινωνίας για διαφορετικές αλληλεπιδράσεις με το Client Software. Το USB μεγιστοποιεί την χρησιμοποίηση του διαύλου διαχωρίζοντας τις διαφορετικές ροές επικοινωνίας μεταξύ του λογισμικού και του function. Κάθε ροή επικοινωνίας τερματίζεται σε ένα Τερματικό Σημείο *(Endpoint)* της συσκευής.

Το Σχήμα 2.2.4 δείχνει με μεγαλύτερη λεπτομέρεια τη μορφή ενός συστήματος USB. Η πραγματική ροή δεδομένων στα στρώματα Device και Function γίνεται μέσω του στρώματος USB Bus Interface. Στο Σχήμα 2.2.4 φαίνεται η πραγματική διαστρωματική ροή δεδομένων και η αφαιρετική παράσταση της ροής δεδομένων σε κάθε στρώμα. Η διαστρωματική επικοινωνία γίνεται μέσω διαφόρων interfaces. Τα interfaces της συσκευής εξαρτώνται από την υλοποίησή της. Στην πλευρά του host υπάρχουν δύο interfaces λογισμικού (από κάτω προς τα πάνω):

- Λογισμικό Οδήγησης του Ελεγκτή του Host *(Host Controller Driver, HCD)*: Το interface λογισμικού μεταξύ του Host Controller του USB και του USB System Software. Αυτό το interface επιτρέπει διαφορετικές υλοποιήσεις του Host Controller, χωρίς να απαιτείται το υπόλοιπο λογισμικό του host να εξαρτάται από τη συγκεκριμένη υλοποίηση. Ο κατασκευαστής του HC παρέχει το κατάλληλο HCD για την οδήγησή της συγκεκριμένης υλοποίησης, το οποίο είναι συμβατό με το Λειτουργικό Σύστημα του Host.

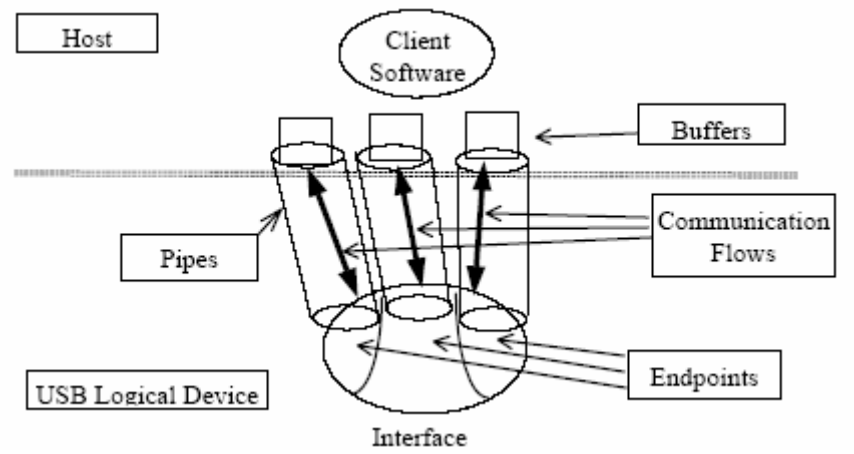
- Λογισμικό Οδήγησης του USB *(USB Driver, USBD)*: Το interface λογισμικού μεταξύ του USB System Software και του Client Software. Αυτό το interface παρέχει στα Client SW με τις απαραίτητες μεθόδους για τον χειρισμό των USB συσκευών.



Σχήμα 2.2.4: Λεπτομερής Περιγραφή της Αλληλεπίδρασης μεταξύ του Host και της Συσκευής

Μια Λογική Συσκευή εμφανίζεται στο USB ως ένα σύνολο από Endpoints. Τα endpoints χωρίζονται σε ομάδες *(Endpoint Sets)* τα οποία υλοποιούν ένα interface της συσκευής. Το function της συσκευής αποτελείται από ένα σύνολο από interfaces. Το USB System Software διαχειρίζεται τη συσκευή χρησιμοποιώντας το Default Control Pipe το οποίο καταλήγει στο endpoint 0 της συσκευής *(Πρωτεύον Pipe Ελέγχου)*. Το Client Software διαχειρίζεται ένα interface χρησιμοποιώντας ένα σύνολο από pipes, το οποίο αναφέρεται ως «Δέσμη από Pipes» *(Pipe Bundle)*, το οποίο σχετίζεται με το συγκεκριμένο Endpoint Set. Το Client Software παράγει αιτήσεις για μεταφορά δεδομένων στο USB μεταξύ ενός buffer στον host και ένα endpoint της συσκευής. Ανάλογα με την κατεύθυνση της μεταφοράς δεδομένων, ο HC ή η USB συσκευή πακετάρει τα δεδομένα για να μεταφερθούν στο USB. Ο HC έχει την ευθύνη για τη χρονική στιγμή που θα πραγματοποιηθεί η μεταφορά δεδομένων.

Στο παρακάτω σχήμα παρουσιάζεται ο τρόπος με τον οποίο μεταφέρονται οι ροές επικοινωνίας *(Communication Flows)* μέσω των pipes, ανάμεσα στα endpoints της συσκευής και των buffers του host. Το σύνολο των ροών επικοινωνίας επιλέγονται από τον κατασκευαστή της συσκευής για την κατάλληλη εξυπηρέτηση της συσκευής.



Σχήμα 2.2.5: Ροή Επικοινωνίας στο USB

Παρακάτω περιγράφονται τα Enpoints, τα Pipes, και οι ροές επικοινωνίας με περισσότερες λεπτομέρειες.

2.2.3.1 Τερματικά Σημεία (Endpoints)

Μια USB συσκευή αποτελείται ένα σύνολο από ανεξάρτητα endpoints. Κάθε endpoint μιας συσκευής είναι μοναδικά αναγνωρίσιμο. Αποτελεί το σημείο όπου τερματίζεται μια ροή επικοινωνίας ανάμεσα στη συσκευή και τον host. Κάθε λογική συσκευή έχει μια μοναδική διεύθυνση στο σύστημα, η οποία δίνεται σε αυτή κατά τη σύνδεση της στον δίαυλο. Σε κάθε endpoint δίνεται ένας μοναδικός και σταθερός αριθμός *(Endpoint Number)* για την διευθυνσιοδότησή του. Κάθε endpoint υποστηρίζει μια κατεύθυνση ροής δεδομένων: Από τον host προς το endpoint *(output)* ή από το endpoint προς τον host *(input)*. Ο συνδυασμός της διεύθυνσης της συσκευής, του αριθμού του endpoint, και της κατεύθυνσής του επιτρέπει σε κάθε endpoint να αναφέρεται με μοναδικό τρόπο στο USB.

Κάθε endpoint έχει κάποια χαρακτηριστικά που καθορίζουν τον τύπο της υπηρεσίας μεταφοράς ανάμεσα σε αυτό και το client software. Αυτά τα χαρακτηριστικά είναι:

- Απαιτήσεις συχνότητας/καθυστέρησης πρόσβασης στον δίαυλο *(Bus Access)*
- Απαιτήσεις σε εύρος ζώνης *(Bandwidth)*
- Ο αριθμός του endpoint
- Απαιτήσεις διαχείρισης λαθών
- Το μέγιστο μέγεθος πακέτου που το endpoint μπορεί να στέλει ή να λάβει
- Ο τύπος Μεταφοράς Δεδομένων που υποστηρίζει (Isochronous, Bulk, Interrupt, Control)

- Η κατεύθυνση που υποστηρίζει

Κάθε συσκευή USB διαθέτει δύο endpoints με αριθμό 0 και αντίθετη κατεύθυνση. Τα endpoints που έχουν αριθμό διάφορο του 0 είναι σε μη καθορισμένη κατάσταση πριν τη ρύθμισή τους και δεν μπορούν να χρησιμοποιηθούν πριν από την κατάλληλη ρύθμιση της συσκευής.

Όλες οι USB συσκευές πρέπει να υλοποιούν μέθοδο ελέγχου που χρησιμοποιεί και τα δύο endpoints 0. Το USB System Software χρησιμοποιεί αυτή τη μέθοδο ελέγχου για να αρχικοποιήσει και γενικά να διαχειριστεί τη λογική συσκευή μέσω του Default Control Pipe. Το pipe αυτό παρέχει πρόσβαση στις πληροφορίες ρύθμισης της συσκευής και επιτρέπει τον γενικό έλεγχο και την παρακολούθηση της κατάστασης *(Status)* της συσκευής. Το Default Control Pipe υποστηρίζει Control Transfers. Τα endpoints 0 είναι πάντα προσπελάσιμα μόλις η συσκευή συνδεθεί.

Τα functions μπορούν να έχουν πρόσθετα endpoints για τη λειτουργία τους. Τα low-speed functions μπορούν να έχουν το πολύ άλλα δύο endpoints. Τα full-speed functions μπορούν να έχουν το πολύ 15 input endpoints και 15 output endpoints.

2.2.3.2 Σωληνώσεις (Pipes)

Τα USB Pipes αποτελούν ένα σύνολο πληροφοριών που παριστούν την ικανότητα μεταφοράς δεδομένων ανάμεσα στο λογισμικό του Host, μέσω ενός buffer στη μνήμη, και ένα endpoint μιας συσκευής. Υπάρχουν δύο αμοιβαία αποκλειόμενα μοντέλα επικοινωνίας με pipes:

- Ροής *(Stream)*: Τα δεδομένα που μεταφέρονται σε ένα pipe δεν έχουν κάποια συγκεκριμένη δομή.

- Μηνυμάτων *(Message)*: Τα δεδομένα που μεταφέρονται σε ένα pipe έχουν συγκεκριμένη δομή.

Το USB δεν ερμηνεύει τα δεδομένα που μεταφέρει. Αυτό γίνεται από το Client Software στην πλευρά του Host, και το function στην πλευρά συσκευής. Κάθε pipe χαρακτηρίζεται από τα εξής:

- Απαιτήσεις στην πρόσβαση στο USB και στο εύρος ζώνης

- Τύπο Μεταφοράς

- Χαρακτηριστικά του endpoint που συμμετέχει στο pipe, όπως κατεύθυνση και μέγιστο μήκος δεδομένων στο πεδίο δεδομένων ενός πακέτου Data *(Maximum Data Payload Size)*. Τα πακέτα USB περιγράφονται στην ενότητα 2.3.

Όπως έχει ήδη περιγραφτεί, όταν η συσκευή συνδεθεί στο USB, το USB System Software χρησιμοποιεί το Default Control Pipe στα endpoints 0 για να αναγνωρίσει τη συσκευή, να διαβάσει τα χαρακτηριστικά της, και να τη ρυθμίσει κατάλληλα. Μετά από αυτή τη διαδικάσία το Default Control Pipe μπορεί να

χρησιμοποιηθεί από το Client SW για άλλους σκοπούς, οι οποίοι ορίζονται από τη συσκευή.

Το Client SW στέλνει αιτήσεις για μεταφορές δεδομένων μέσω των Πακέτων Αιτήσεων Εισόδου/Εξόδου *(I/O Request Packets, IRPs)*. Ο ορισμός των IRP εξαρτάται από το λειτουργικό σύστημα. Σε κάθε περίπτωση, τα IRP πρέπει να παρέχουν όλες τις μεταβλητές που είναι απαραίτητες για τον πλήρη καθορισμό των παραμέτρων της μεταφοράς δεδομένων. Το Client SW πληροφορείται ότι ένα IRP έχει ολοκληρωθεί όταν η συναλλαγή *(transaction)* στον δίαυλο που σχετίζεται με αυτό ολοκληρωθεί, είτε επιτυχώς, είτε λόγω σφαλμάτων.

Η παρουσία εκκρεμόντων IRP για ένα pipe έχει ως αποτέλεσμα τη μεταφορά δεδομένων στον δίαυλο. Αν δεν υπάρχουν IRP που εκκρεμούν ή σε εξέλιξη για ένα pipe, το pipe παραμένει ανενεργό και ο HC δεν θα κάνει καμία ενέργεια που σχετίζεται με αυτό. Εάν σε ένα μη isochronous pipe δεν μπορεί να γίνει η μεταφορά δεδομένων *(κατάσταση STALL)* ή το όριο σφαλμάτων για ένα πακέτο ξεπεραστεί (βλέπε Ενότητα 2.3) το αντίστοιχο IRP θα αποσυρθεί και δεν θα γίνονται δεκτά άλλα IRP για αυτό το pipe, μέχρι το Client SW να αντιμετωπίσει το σφάλμα. Η μέθοδος που αντιμετωπίζεται το σφάλμα εξαρτάται από την υλοποίηση του Client SW. Τα IRP που ολοκληρώνονται επιτυχώς αποσύρονται και η ροή των IRP συνεχίζεται κανονικά.

Όταν ένα IRP εκτελείται, το μέγεθος δεδομένων (*Data Payload*) κάθε πακέτου ισούται με το μέγιστο μέγεθος πακέτου, εκτός από το τελευταίο πακέτο που περιέχει ότι έχει απομείνει από τα δεδομένα του IRP. Τα πακέτα που έχουν payload μικρότερο από το μέγιστο *(Short Packets)* ερμηνεύονται ανάλογα με αυτό που αναμένει το Client SW ως εξής:

- Το Client SW αναμένει δεδομένα μεταβλητού μήκους σε ένα IRP. Στην περίπτωση αυτή το short packet θεωρείται σωστό και σημαίνει το τέλος τον δεδομένων. Το IRP αποσύρεται χωρίς σφάλματα.

- Το Client SW αναμένει δεδομένα συγκεκριμένου μήκους σε ένα IRP. Στην περίπτωση αυτή το short packet θεωρείται λανθασμένο και προκαλείται σφάλμα. Το IRP αποσύρεται, όπως και όλα τα IRP που σχετίζονται με αυτό το pipe, και το pipe μπλοκάρεται.

Τα IRP πρέπει να αναφέρουν ποια συμπεριφορά από τις παραπάνω θα ακολουθηθεί στην περίπτωση ανίχνευσης ενός short packet από τον HC.

2.2.3.2.1 Σωληνώσεις Ροής (Stream Pipes)

Τα Stream Pipes μεταφέρουν τα δεδομένα χωρίς κάποια δομή που να ορίζεται από το USB για το περιεχόμενο των δεδομένων. Τα δεδομένα φτάνουν στον δέκτη με την ίδια σειρά που στέλνονται από τον πομπό. Τα stream pipes είναι πάντα μιας κατεύθυνσης. Το USB δεν παρέχει συγχρονισμό σε περίπτωση που ένα stream pipe χρησιμοποιείται από περισσότερα από ένα Clients. Σε κάθε περίπτωση το USB λειτουργεί σαν να υπάρχει ένα μόνο client.

Ένα stream pipe χρησιμοποιεί ένα endpoint κατάλληλης κατεύθυνσης της συσκευής. Το endpoint με τον ίδιο αριθμό και αντίθετης κατεύθυνσης μπορεί να χρησιμοποιηθεί για ένα άλλο stream pipe. Τα stream pipes υποστηρίζουν isochronous, interrupt και bulk transfers.

2.2.3.2.2 Σωληνώσεις Μηνυμάτων (Message Pipes)

Τα Message Pipes έχουν συγκεκριμένη δομή ροής της επικοινωνίας. Αρχικά, στέλνεται μια αίτηση από τον Host προς τη συσκευή. Ακολουθεί μία ή περισσότερες μεταφορές δεδομένων προς την κατάλληλη κατεύθυνση. Τέλος, ακολουθεί ένα στάδιο αναφοράς κατάστασης *(Status Stage)*. Ο τρόπος αυτός επικοινωνίας εξασφαλίζει ότι οι εντολές που στέλνονται μέσω των message pipes ερμηνεύονται σωστά. Ένα message pipe επιτρέπει επικοινωνία και προς τις δύο κατευθύνσεις.

Το USB System Software πρέπει να εξασφαλίζει ότι σε κάθε στιγμή μόνο ένα Client SW χρησιμοποιεί ένα συγκεκριμένο message pipe. Σε κάθε περίπτωση οι αιτήσεις μεταφέρονται μέσω του pipe σε ουρά FIFO *(First-in, First-out)*.

Ένα message pipe δεν στέλνει το επόμενο μήνυμα από τον host μέχρι την ολοκλήρωση της επεξεργασίας του τρέχοντος μηνύματος από τη συσκευή. Όμως υπάρχουν καταστάσεις σφάλματος, στις οποίες μια μεταφορά εγκαταλείπεται από τον host και το message pipe στέλνει ένα μήνυμα πριν από τη στιγμή που αναμένεται από τη συσκευή. Από την πλευρά του λογισμικού του host, το IRP στο οποίο εντοπίστηκε το σφάλμα αποσύρεται, όπως και όλα τα IRP που σχετίζονται με το συγκεκριμένο pipe.

Ένα message pipe απαιτεί έναν αριθμό endpoint και προς τις δύο κατευθύνσεις. Το USB δεν επιτρέπει ένα message pipe να σχετίζεται με δύο διαφορετικούς αριθμούς endpoints για κάθε κατεύθυνση. Τα message pipes υποστηρίζουν control transfers.

2.2.4 Τύποι Μεταφοράς Δεδομένων

Το USB χρησιμοποιεί τα pipes για τη μεταφορά δεδομένων από τα buffers του host στα functions των συσκευών. Για την καλύτερη εξυπηρέτηση των απαιτήσεων των συσκευών, το USB υποστηρίζει διάφορους τύπους μεταφοράς *(Transfer Types)*. Ο τύπος μεταφοράς που χρησιμοποιείται για τη μεταφορά δεδομένων σε ένα pipe ανάμεσα σε ένα endpoint μιας συσκευής και το αντίστοιχο client SW καθορίζεται από τα χαρακτηριστικά του endpoint. Κάθε τύπος μεταφοράς καθορίζει διάφορα χαρακτηριστικά της ροής δεδομένων, όπως:

- Το σημασιολογικό περιεχόμενο των δεδομένων
- Τη μορφή κάθε συναλλαγής *(transaction)*
- Την κατεύθυνση της ροής δεδομένων

- Περιορισμούς στα μεγέθη των πακέτων
- Περιορισμούς καθυστέρησης
- Απαιτήσεις στην πρόσβαση στον δίαυλο
- Τον τρόπο χειρισμού των σφαλμάτων

Τα χαρακτηριστικά αυτά παραμένουν σταθερά από τη στιγμή που εγκαθίσταται το pipe μέχρι τον τερματισμό του. Το υπόλοιπο της ενότητας περιγράφει τις ιδότητες των τύπων μεταφοράς δεδομένων: Control, Isochronous, Interrupt και Bulk.

2.2.5 Μεταφορές Ελέγχου (Control Transfers)

Οι Μεταφορές Ελέγχου είναι μη περιοδικές και χρησιμοποιούνται για τη ροή πληροφοριών ρυθμίσης *(Configuration)* και κατάστασης *(Status)*, και εντολών *(Commands)* μεταξύ του client SW και του αντίστοιχου function. Οι control transfers πραγματοποιούνται μόνο μέσω message pipes. Η πιο χαρακτηριστική περίπτωση control transfer είναι η μεταφορά δεδομένων στο default control pipe μιας συσκευής, όπου διαβάζονται από τον host όλες οι απαραίτητες πληροφορίες της συσκευής για τον έλεγχο, τη ρύθμιση και την κατάσταση της συσκευής.

Εφόσον οι μεταφορές ελέγχου πραγματοποιούνται μέσω message pipes, η δομή τους είναι καθορισμένη. Μια control transfer πραγματοποιείται σε τρία στάδια, όπως απαιτούν τα message pipes: Εγκατάστασης *(Setup)*, Δεδομένων *(Data)*, και Κατάστασης *(Status)*. Στο στάδιο εγκατάστασης ο host στέλνει μια αίτηση στο function. Η αίτηση αυτή είναι μια εντολή, από το σύνολο των εντολών του USB, του host προς το function. Στο στάδιο δεδομένων γίνεται μεταφορά δεδομένων προς την κατεύθυνση που έχει υποδειχθεί κατά το στάδιο εγκατάστασης. Το στάδιο αυτό μπορεί να απουσιάζει. Τέλος, στο στάδιο κατάστασης στέλνονται δεδομένα που αναφέρουν την κατάσταση της συσκευής ή του Host μετά από τις ενέργειες που έγιναν. Τα στάδια εγκατάστασης και κατάστασης πραγματοποιούνται με ένα transaction το καθένα. Το στάδιο δεδομένων μπορεί να πραγματοποιείται με ένα ή περισσότερα transactions.

Οι control transfers είναι διπλής κατεύθυνσης και χρησιμοποιούν το input και output endpoint του συγκεκριμένου αριθμού endpoint που συμμετέχει στο message pipe. Απαιτείται ανίχνευση και διόρθωση λαθών. Στην Παράγραφο 2.3.4.2 περιγράφεται λεπτομερώς η μορφή των control transfers.

2.2.5.1 Περιορισμοί Μεγέθους Πακέτων

Ένα endpoint για control transfers καθορίζει το μέγιστο μήκος δεδομένων *(Maximum Data Payload Size)* των πακέτων δεδομένων που μπορεί να στείλει ή να λάβει το συγκεκριμένο endpoint. Σε αυτό δεν περιλαμβάνονται τα bytes πρωτοκόλλου των πακέτων αλλά μόνο τα bytes δεδομένων.

Για full-speed συσκεύες το μέγιστο data payload μπορεί να είναι 8, 16, 32, ή 64 bytes, ενώ για low-speed συσκευές είναι πάντα 8 bytes. Όλοι οι Host Controllers πρέπει να υποστηρίζουν τα παραπάνω μέγιστα data payloads. Ένα endpoint πρέπει να αναφέρει στις πληροφορίες ρύθμισής του το μέγιστο data payload που υποστηρίζει. Το USB System Software διαβάζει το μέγιστο data payload όλων των endpoints μιας συσκευής μέσω του default control pipe και στη συνέχεια χρησιμοποιεί την κατάλληλη τιμή για την επικοινωνία με κάθε endpoint.

Κατά την επικοινωνία του host με ένα endpoint επιτρέπεται η μετάδοση πακέτων δεδομένων με data payload μικρότερο ή ίσο με το μέγιστου data payload, αλλιώς όλα τα IRP του pipe όπου παρατηρείται το σφάλμα αποσύρονται. Όταν το μήκος των δεδομένων που υποδεικνύει ένα IRP είναι μεγαλύτερο από το μέγιστο data payload, τότε όλα τα πακέτα έχουν data payload ίσο με το μέγιστο, εκτός από το τελευταίο πακέτο που περιέχει τα υπόλοιπα δεδομένα. Εάν το μήκος των δεδομένων είναι ακέραιο πολλαπλάσιο του μέγιστου data payload, τότε μετά τη μετάδοση του τελευταίου πακέτου δεδομένων ακολουθεί ένα πακέτο δεδομένων μηδενικού data payload.

2.2.5.2 Απαιτήσεις Πρόσβασης στον Δίαυλο

Ένα Control Endpoint δεν μπορεί να υποδείξει την επιθυμητή συχνότητα πρόσβασης στον δίαυλο ενός control pipe. Το USB κάνει την «καλύτερη προσπάθεια» *("Best Effort")* για να εξυπηρετήσει τα εκκρεμή IRP για control transfers με βάση τις απαιτήσεις κάθε message pipe. Το USB χρησιμοποιεί τους ακόλουθους κανόνες για την χρονοδρομολλόγηση των control transfers:

- Το USB χρησιμοποιεί το 10 % του χρόνου ενός frame για control transfers. Εάν κάποιο μέρος αυτού του χρόνου μείνει ανεκμετάλλευτο από τα control transfers, τότε μπορεί να χρησιμοποιηθεί για bulk transfers.

- Ένα control transfer που αποτυγχάνει και πρέπει να επαναληφθεί μπορεί να επαναληφθεί στο τρέχον ή σε επόμενο frame.

- Εάν υπάρχει ανεκμετάλλευτος χρόνος στο frame, που είχε αρχικά δεσμευτεί για isochronous ή interrupt transfers, μπορεί να χρησιμοποιηθεί για control transfers που δεν έχουν εξυπηρετηθεί.

- Ο Host Controller θα πρέπει να εξυπηρετεί με «δικαιοσύνη» τα control transfers για κάθε endpoint. Ο τρόπος με τον οποίο επιτυγχάνεται αυτό εξαρτάται από την υλοποίηση.

- Εάν τα εκκρεμή IRP για control transfers δεν χωράνε όλα σε ένα frame, τότε θα εξυπηρετηθούν με βάση τους παραπάνω κανόνες στο τρέχον και τα επόμενα frames.

Από τα παραπάνω προκύπτει ότι δεν υπάρχει ένας συγκεκριμένος ρυθμός εξυπηρέτησης των control transfers ενός endpoint. Στους πίνακες 2.3.1 και 2.3.2 παρουσιάζεται ο μέγιστος αριθμός control transfers που μπορεί να εξυπηρετηθεί για μια δεδομένη τιμή data payload των πακέτων δεδομένων σε ένα frame για full-speed

και low-speed συσκευές αντίστοιχα, υποθέτωντας ότι υπάρχουν αιτήσεις μόνο για control transfers και επομένως χρησιμοποιείται όλος ο χρόνος του frame για την εξυπηρέτησή τους. Επίσης υποτίθεται ότι το στάδιο δεδομένων κάθε control transfer πραγματοποιείται με την αποστολή ενός πακέτου δεδομένων, και ότι δεν αναμένεται χειραψία από τον δέκτη των δεδομένων. Ένας Host Controller μπορεί να μην δύναται να δώσει το θεωρητικό μέγιστο αριθμό μεταφορών ελέγχου σε ένα frame για διάφορους λόγους (σφάλματα, καθυστερήσεις, λόγοι υλοποίησης).

Πίνακας 2.2.1: Μέγιστος αριθμός μεταφορών Ελέγχου συναρτήσει του μεγέθους των πακέτων δεδομένων για full-speed συσκευές

45 Bytes Πρωτοκόλλου		9 SYNC bytes, 9 PID bytes, 6 Endpoint + CRC bytes, 6 CRC bytes, 8 Setup data bytes, και καθυστέρηση 7 bytes μεταξύ των πακέτων, βλέπε Ενότητα 2.3			
Data Payload	**Μέγιστο Εύρος Ζώνης (bytes/sec)**	**% Εύρος Ζώνης του Frame ανά Μεταφορά**	**Μέγιστος Αριθμός Μεταφορών**	**Υπολοιπόμενα Bytes**	**Bytes Δεδομένων/Frame**
1	32000	3 %	32	23	32
2	62000	3 %	31	43	62
4	120000	3 %	30	30	120
8	224000	4 %	28	16	224
16	384000	4 %	24	36	384
32	608000	5 %	19	37	608
64	832000	7 %	13	83	832
Max	1500000				1500

Πίνακας 2.2.2: Μέγιστος αριθμός μεταφορών Ελέγχου συναρτήσει του μεγέθους των πακέτων δεδομένων για low-speed συσκευές

46 Bytes Πρωτοκόλλου					
Data Payload	**Μέγιστο Εύρος Ζώνης (bytes/sec)**	**% Εύρος Ζώνης του Frame ανά Μεταφορά**	**Μέγιστος Αριθμός Μεταφορών**	**Υπολοιπόμενα Bytes**	**Bytes Δεδομένων/Frame**
1	3000	25 %	3	46	3
2	6000	26 %	3	43	6
4	12000	29 %	3	37	12
8	24000	4 %	3	25	24
Max	187500				187

2.2.6 Ισόχρονες Μεταφορές (Isochronous Transfers)

Οι Ισόχρονες μεταφορές απαιτούν περιοδική και συνεχή επικοινωνία μεταξύ του Host και μιας συσκευής. Συνήθως τα δεδομένα που μεταφέρονται χρησιμοποιούνται για εφαρμογές πραγματικού χρόνου. Γενικά όμως, οι ισόχρονες μεταφορές μπορούν να χρησιμοποιηθούν για οποιαδήποτε μεταφορά δεδομένων η οποία είναι ανεκτική σε σφάλματα. Οι ισόχρονες μεταφορές πραγματοποιούνται μέσω stream pipes και χρησιμοποιούνται για την επικοινωνία μόνο full-speed συσκευών. Οι ισόχρονες μεταφορές έχουν τα εξής χαρακτηριστικά:

- Εγγυημένη πρόσβαση στον δίαυλο με περιορισμένη καθυστέρηση.
- Εγγυημένα σταθερό ρυθμό μετάδοσης στο pipe, για όσο χρόνο υπάρχουν αιτήσεις για μετάδοση μέσω του pipe.
- Σε περίπτωση που υπάρχει σφάλμα στην μεταφορά δεδομένων, δεν γίνεται επανάληψη της μεταφοράς.

Εφόσον οι ισόχρονες μεταφορές πραγματοποιούνται μέσω stream pipes, δεν έχουν κάποια συγκεκριμένη σημασιολογική δομή πρωτοκόλλου. Τα δεδομένα παραδίδονται με διαδοχικά transactions. Ο δέκτης μπορεί να ανιχνεύσει τα σφάλματα που μπορεί να προκύψουν και να στείλει τη σωστή χειραψία στον πομπό. Σε κάθε περίπτωση ο πομπός θα αγνοήσει τη χειραψία και η διαδικασία θα συνεχιστεί με το επόμενο transaction.

Οι ισόχρονες μεταφορές είναι μιας κατεύθυνσης αφού χρησιμοποιούν stream pipes. Η κατεύθυνση της μεταφοράς καθορίζεται από την κατεύθυνση του endpoint *(input ή output)* που χρησιμοποιεί το pipe. Εάν μια συσκευή απαιτεί ισόχρονη μεταφορά δεδομένων διπλής κατεύθυνσης, τότε θα χρησιμοποιηθούν δύο stream pipes (δύο endpoints διαφορετικής κατεύθυνσης), ένα για κάθε κατεύθυνση. Στην Παράγραφο 2.3.4.4 περιγράφεται λεπτομερώς η μορφή των isochronous transfers.

2.2.6.1 Περιορισμοί Μεγέθους Πακέτων

Κάθε isochronous endpoint καθορίζει το μέγιστο data payload που μπορεί να λάβει ή να στείλει. Ο host διαβάζει το μέγιστο data payload που υποστηρίζει ένα isochronous endpoint κατά τη ρύθμιση της συσκευής, μέσω του default control pipe, και ελέγχει αν υπάρχει αρκετός χρόνος στον διάυλο για να χωρέσει αυτό το data payload σε κάθε frame. Εάν υπάρχει αρκετός χρόνος στον δίαυλο το endpoint θα υποστηριχθεί από το USB, αλλιώς όχι. Αυτό έρχεται σε αντίθεση με τις μεταφορές ελέγχου, όπου το USB System Software προσαρμόζει το data payload των πακέτων δεδομένων σύμφωνα με τις απαιτήσεις του endpoint.

Το μέγιστο data payload που υποστηρίζει το USB για ισόχρονες μεταφορές είναι 1023 bytes ή μικρότερο. Στον Πίνακα 2.2.3 παρουσιάζονται πληροφορίες για διαφορετικά μεγέθη των isochronous transactions και ο μέγιστος αριθμός των transactions σε ένα frame για κάθε μέγεθος, θεωρώντας ότι όλος ο χρόνος του frame χρησιμοποιείται για isochronous transfers. Ένας Host Controller μπορεί να μην

δύναται να δώσει το θεωρητικό μέγιστο αριθμό μεταφορών ελέγχου σε ένα frame για διάφορους λόγους (σφάλματα, καθυστερήσεις, λόγοι υλοποίησης).

Κάθε transaction ενός isochronous pipe πρέπει να έχει data payload μικρότερο ή ίσο από το μέγιστο, αλλιώς όλα τα IRP του pipe αποσύρονται. Το data payload καθορίζεται από τον πομπό (Client SW ή function). Το USB εγγυάται ότι οποιοδήποτε επιτρπτό μέγεθος και αν χρησιμοποιείται, το πακέτο θα μεταδοθεί. Διάφορα σφάλματα μπορεί να αλλιώσουν το πραγματικό μέγεθος του πακέτου που φτάνει στον δέκτη. Ο δέκτης μπορεί να αγνοήσει το λανθασμένο πακέτο, είτε ανιχνεύοντας τα σφάλματα, είτε συγκρίνοντας το μέγεθος του πακέτου με το αναμενόμενο μέγεθος.

Πίνακας 2.2.3: Μέγιστος αριθμός Ισόχρονων μεταφορών συναρτήσει του μεγέθους των πακέτων δεδομένων για full-speed συσκευές

9 Bytes Πρωτοκόλλου		2 SYNC bytes, 2 PID bytes, 2 Endpoint + CRC bytes, 2 CRC bytes, και καθυστέρηση 1 byte μεταξύ των πακέτων, βλέπε Ενότητα 2.3			
Data Payload	**Μέγιστο Εύρος Ζώνης (bytes/sec)**	**% Εύρος Ζώνης του Frame ανά Μεταφορά**	**Μέγιστος Αριθμός Μεταφορών**	**Υπολοιπόμενα Bytes**	**Bytes Δεδομένων/Frame**
1	150000	1 %	150	0	150
2	272000	1 %	136	4	272
4	460000	1 %	115	5	460
8	704000	1 %	88	4	704
16	960000	2 %	60	0	960
32	1152000	3 %	36	24	1152
64	1280000	5 %	20	40	1280
128	1280000	9 %	10	130	1280
256	1280000	18 %	5	175	1280
512	1024000	35 %	2	458	1024
1023	1023000	69 %	1	458	1023
Max	1500000				1500

2.2.6.2 Απαιτήσεις Πρόσβασης στον Δίαυλο

Οι ισόχρονες μεταφορές χρησιμοποιούνται μόνο από full-speed συσκευές. Το USB ορίζει ότι το πολύ 90 % του χρόνου ενός frame μπορεί να χρησιμοποιηθεί για isochronous και interrupt transfers, με προτεραιότητα στις πρώτες. Αν μείνει αναξιοποίητος χρόνος μπορεί να χρησιμοποιηθεί για bulk transfers.

Κάθε isochronous pipe μπορεί να μεταφέρει μόνο ένα πακέτο δεδομένων σε κάθε frame. Σφάλματα στον δίαυλο ή καθυστερήσεις στο λειτουργικό σύστημα μπορεί να έχουν ως αποτέλεσμα την αποτυχία μετάδοσης ενός πακέτου σε ένα frame. Το client SW πρέπει να ενημερώνεται για αυτή την κατάσταση. Επίσης η συσκευή

που αναμένει τα δεδομένα μπορεί να ανιχνεύσει αυτή την κατάσταση ανιχνεύοντας δύο διαδοχικά πακέτα SOF στον δίαυλο χωρίς την παρουσία δεδομένων για ένα isochronous endpoint της συσκευής ανάμεσα στην λήψη των SOF.

2.2.7 Μεταφορές Διακοπής (Interrupt Transfers)

Οι μεταφορές Διακοπής είναι σχεδιασμένες για την υποστήριξη συσκευών που απαιτούν αποστολή ή λήψη μικρών ποσοτήτων δεδομένων κατά περιοδικές αλλά όχι συχνές χρονικές στιγμές. Οι μεταφορές διακοπής χρησιμοποιούν stream pipes και εφαρμόζονται σε low-speed και full-speed συσκευές. Οι μεταφορές διακοπής έχουν τα εξής χαρακτηριστικά:

- Εγγυημένη εξυπηρέτηση των συσκευών κατά τις περιοδικές χρονικές στιγμές που απαιτείται μεταφορά δεδομένων.

- Επανάληψη μιας μεταφοράς δεδομένων στην επόμενη περίοδο εξυπηρέτησης, στην περίπτωση σφάλματος στον δίαυλο κατά τη μεταφορά.

Εφόσον οι μεταφορές διακοπής πραγματοποιούνται μέσω stream pipes, δεν έχουν κάποια συγκεκριμένη σημασιολογική δομή πρωτοκόλλου. Τα δεδομένα παραδίδονται κατά τις περιόδους εξυπηρέτησης και τα δεδομένα που λαμβάνονται από τον δέκτη ελέγχονται για σφάλματα. Ο δέκτης πρέπει να απαντήσει στον πομπό με μια κατάλληλη χειραψία.

Οι μεταφορές διακοπής είναι μιας κατεύθυνσης αφού χρησιμοποιούν stream pipes. Η κατεύθυνση της μεταφοράς καθορίζεται από την κατεύθυνση του endpoint *(input ή output)* που χρησιμοποιεί το pipe. Στην Παράγραφο 2.3.4.3 περιγράφεται λεπτομερώς η μορφή των interrupt transfers.

2.2.7.1 Περιορισμοί Μεγέθους Πακέτων

Κάθε interrupt endpoint καθορίζει το μέγιστο data payload που μπορεί να λάβει ή να στείλει. Το μέγιστο data payload που υποστηρίζει το USB για μεταφορές διακοπής είναι 64 bytes ή μικρότερο για τις full-speed συσκευές, και 8 bytes ή μικρότερο για τις low-speed συσκευές. Όλοι οι Host Controllers υλοποιούνται έτσι ώστε να υποστηρίζουν τα παραπάνω μέγιστα data payloads.

Ο host διαβάζει το μέγιστο data payload που υποστηρίζει ένα interrupt endpoint κατά τη ρύθμιση της συσκευής, μέσω του default control pipe, και ελέγχει αν υπάρχει αρκετός χρόνος στον διάυλο για να χωρέσει αυτό το data payload σε κάθε frame. Εάν υπάρχει αρκετός χρόνος στον δίαυλο το endpoint θα υποστηριχθεί από το USB, αλλιώς όχι.

Κάθε transaction ενός interrupt pipe πρέπει να έχει data payload μικρότερο ή ίσο από το μέγιστο, αλλιώς το όλα τα IRP του pipe αποσύρονται. Το data payload καθορίζεται από τον πομπό (Client SW ή function). Το USB εγγυάται ότι οποιοδήποτε επιτρεπτό μέγεθος και αν χρησιμοποιείται, το πακέτο θα μεταδοθεί.

Όταν το μήκος των δεδομένων που υποδεικνύει ένα IRP είναι μεγαλύτερο από το μέγιστο data payload, τότε όλα τα πακέτα έχουν data payload ίσο με το μέγιστο, εκτός από το τελευταίο πακέτο που περιέχει τα υπόλοιπα δεδομένα. Εάν το μήκος των δεδομένων είναι ακέραιο πολλαπλάσιο του μέγιστου data payload, τότε μετά τη μετάδοση του τελευταίου πακέτου δεδομένων ακολουθεί ένα πακέτο δεδομένων μηδενικού data payload.

Στους Πίνακες 2.2.4 και 2.2.5 παρουσιάζονται πληροφορίες για διαφορετικά μεγέθη των interrupt transactions και ο μέγιστος αριθμός των transactions σε ένα frame για κάθε μέγεθος, θεωρώντας ότι όλος ο χρόνος του frame χρησιμοποιείται για interrupt transfers, για full-speed και low-speed συσκευές αντίστοιχα. Ένας Host Controller μπορεί να μην δύναται να δώσει το θεωρητικό μέγιστο αριθμό μεταφορών ελέγχου σε ένα frame για διάφορους λόγους (σφάλματα, καθυστερήσεις, λόγοι υλοποίησης)

Πίνακας 2.2.4: Μέγιστος αριθμός μεταφορών Διακοπής συναρτήσει του μεγέθους των πακέτων δεδομένων για full-speed συσκευές

13 Bytes Πρωτοκόλλου		3 SYNC bytes, 3 PID bytes, 2 Endpoint + CRC bytes, 2 CRC bytes, και καθυστέρηση 3 bytes μεταξύ των πακέτων, βλέπε Ενότητα 2.3			
Data Payload	**Μέγιστο Εύρος Ζώνης (bytes/sec)**	**% Εύρος Ζώνης του Frame ανά Μεταφορά**	**Μέγιστος Αριθμός Μεταφορών**	**Υπολοιπόμενα Bytes**	**Bytes Δεδομένων/Frame**
1	107000	1 %	107	2	107
2	200000	1 %	100	0	200
4	352000	1 %	88	4	352
8	568000	1 %	71	9	568
16	816000	2 %	51	21	816
32	1056000	3 %	33	15	1056
64	1216000	5 %	19	37	1216
Max	1500000				1500

Πίνακας 2.2.5: Μέγιστος αριθμός μεταφορών Διακοπής συναρτήσει του μεγέθους των πακέτων δεδομένων για low-speed συσκευές

13 Bytes Πρωτοκόλλου					
Data Payload	**Μέγιστο Εύρος Ζώνης (bytes/sec)**	**% Εύρος Ζώνης του Frame ανά Μεταφορά**	**Μέγιστος Αριθμός Μεταφορών**	**Υπολοιπόμενα Bytes**	**Bytes Δεδομένων/Frame**
1	13000	7 %	13	5	13
2	24000	8 %	12	7	24
4	44000	9 %	11	0	44

8	64000	11 %	8	19	64
Max	187500				187

2.2.7.2 Απαιτήσεις Πρόσβασης στον Δίαυλο

Το USB ορίζει ότι το πολύ 90 % του χρόνου ενός frame μπορεί να χρησιμοποιηθεί για isochronous και interrupt transfers, με προτεραιότητα στις πρώτες. Αν μείνει αναξιοποίητος χρόνος μπορεί να χρησιμοποιηθεί για bulk transfers

Τα interrupt endpoints δηλώνουν σαφώς τις απαιτήσεις τους στην πρόσβαση στον δίαυλο κατά τη ρύθμιση της συσκευής, μέσω του deafult control pipe. Η περίοδος εξυπηρέτησης ενός endpoint μπορεί να είναι από 1 ms μέχρι 255 ms. Το USB System Software χρησιμοποιεί αυτή την πληροφορία και πρέπει να εξασφαλίζει ότι θα εξυπηρετεί περιοδικά *(Polling)* ένα interrupt endpoint σε χρόνους μικρότερους ή ίσους με την περίοδο εξυπηρέτησής του. Το λογισμικό δεν πρέπει εξυπηρετεί κανένα endpoint σε χρόνους μικρότερους από 1 ms. Σε κάθε περίπτωση ένα interrupt endpoint εξυπηρετείται όταν εκκρεμεί ένα σχετικό IRP. Για λόγους σφαλμάτων στον δίαυλο ή καθυστερήσεων στο λειτουργικό σύστημα μπορεί ένα endpoint να μην εξυπηρετηθεί την κατάλληλη στιγμή. Τα δεδομένα του endpoint θα μεταφερθούν στην επόμενη περίοδο εξυπηρέτησης.

Στην περίπτωση ενός input endpoint, ο μόνος τρόπος για να καθορίσει ο host εάν υπάρχουν δεδομένα στο endpoint για αποστολή, δηλαδή έχει συμβεί διακοπή στο endpoint, είναι να του στείλει μια αίτηση για αποστολή δεδομένων προς τον host. Εάν το endpoint δεν έχει δεδομένα για αποστολή τότε απαντά στον host με μια κατάλληλη χειραψία, αλλιώς στέλνει τα δεδομένα.

2.2.8 Ογκώδεις Μεταφορές (Bulk Transfers)

Οι Ογκώδεις μεταφορές είναι σχεδιασμένες για την υποστήριξη συσκευών που στέλνουν ή λαμβάνουν μεγάλες ποσότητες δεδομένων για τις οποίες η καθυστέρηση αποστολής ή λήψης δεν είναι σημαντικός παράγοντας. Οι ογκώδεις μεταφορές χρησιμοποιούν stream pipes και χρησιμοποιούνται μόνο από full-speed συσκευές. Οι ογκώδεις μεταφορές έχουν τα εξής χαρακτηριστικά:

- Αποκτούν πρόσβαση στον δίαυλο όποτε υπάρχει ελεύθερο εύρος ζώνης, δηλαδή όταν η κίνηση στον δίαυλο από τα υπόλοιπα είδη μεταφορών δεν καλύπτει όλο το χρόνο ενός frame.

- Επανάληψη μιας μεταφοράς δεδομένων, στην περίπτωση σφάλματος στον δίαυλο.

- Εγγυημένη παράδοση δεδομένων, αλλά όχι εγγυημένο εύρος ζώνης ή καθυστέρηση.

Εφόσον οι ογκώδεις μεταφορές πραγματοποιούνται μέσω stream pipes, δεν έχουν κάποια συγκεκριμένη σημασιολογική δομή πρωτοκόλλου. Ο δέκτης ελέγχει τα

πακέτα δεδομένων που λαμβάνει για σφάλματα και απαντά στον πομπό με μια κατάλληλη χειραψία. Οι ογκώδεις μεταφορές είναι μιας κατεύθυνσης αφού χρησιμοποιούν stream pipes. Η κατεύθυνση της μεταφοράς καθορίζεται από την κατεύθυνση του endpoint *(input ή output)* που χρησιμοποιεί το pipe. Στην Παράγραφο 2.3.4.1 περιγράφεται λεπτομερώς η μορφή των bulk transfers.

2.2.8.1 Περιορισμοί Μεγέθους Πακέτων

Κάθε bulk endpoint καθορίζει το μέγιστο data payload που μπορεί να λάβει ή να στείλει. Το μέγιστο data payload που υποστηρίζει το USB για ογκώδεις μεταφορές μπορεί να είναι 8, 16, 32 ή 64 bytes. Όλοι οι Host Controllers υλοποιούνται έτσι ώστε να υποστηρίζουν τα παραπάνω μέγιστα data payloads.

Ο host διαβάζει το μέγιστο data payload που υποστηρίζει ένα bulk endpoint κατά τη ρύθμιση της συσκευής, και το χρησιμοποιεί για τις επόμενες μεταφορές δεδομένων προς αυτό το endpoint.

Κάθε transaction ενός bulk pipe πρέπει να έχει data payload μικρότερο ή ίσο από το μέγιστο, αλλιώς όλα τα IRP του pipe αποσύρονται. Το data payload καθορίζεται από τον πομπό (Client SW ή function). Το USB εγγυάται ότι οποιοδήποτε επιτρεπτό μέγεθος και αν χρησιμοποιείται, το πακέτο θα μεταδοθεί. Όταν το μήκος των δεδομένων που υποδεικνύει ένα IRP είναι μεγαλύτερο από το μέγιστο data payload, τότε όλα τα πακέτα έχουν data payload ίσο με το μέγιστο, εκτός από το τελευταίο πακέτο που περιέχει τα υπόλοιπα δεδομένα. Εάν το μήκος των δεδομένων είναι ακέραιο πολλαπλάσιο του μέγιστου data payload, τότε μετά τη μετάδοση του τελευταίου πακέτου δεδομένων ακολουθεί ένα πακέτο δεδομένων μηδενικού data payload.

Στον Πίνακα 2.2.6 παρουσιάζονται πληροφορίες για διαφορετικά μεγέθη των bulk transactions και ο μέγιστος αριθμός των transactions σε ένα frame για κάθε μέγεθος, θεωρώντας ότι όλος ο χρόνος του frame χρησιμοποιείται για bulk transfers. Ένας Host Controller μπορεί να μην δύναται να δώσει το θεωρητικό μέγιστο αριθμό μεταφορών ελέγχου σε ένα frame για διάφορους λόγους (σφάλματα, καθυστερήσεις, λόγοι υλοποίησης)

Πίνακας 2.2.6: Μέγιστος αριθμός Ογκώδων μεταφορών συναρτήσει του μεγέθους των πακέτων δεδομένων για full-speed συσκευές

13 Bytes Πρωτοκόλλου		3 SYNC bytes, 3 PID bytes, 2 Endpoint + CRC bytes, 2 CRC bytes, και καθυστέρηση 3 bytes μεταξύ των πακέτων, βλέπε Ενότητα 2.3			
Data Payload	**Μέγιστο Εύρος Ζώνης (bytes/sec)**	**% Εύρος Ζώνης του Frame ανά Μεταφορά**	**Μέγιστος Αριθμός Μεταφορών**	**Υπολοιπόμενα Bytes**	**Bytes Δεδομένων/Frame**
8	568000	1 %	71	9	568

16	816000	2 %	51	21	816
32	1056000	3 %	33	15	1056
64	1216000	5 %	19	37	1216
Max	1500000				1500

2.2.8.2 Απαιτήσεις Πρόσβασης στον Δίαυλο

Οι bulk transfers χρησιμοποιούνται μόνο για full-speed συσκευές. Τα bulk endpoints δεν μπορούν να υποδείξουν στο USB κάποια επιθυμητή συχνότητα πρόσβασης στον δίαυλο. Τα bulk transfers έχουν την μικρότερη προτεραιότητα πρόσβασης στον δίαυλο από όλους τους άλλους τύπους μεταφοράς. Πραγματοποιούνται μόνο όταν υπάρχει έλευθερος χρόνος σε ένα frame, χωρίς αυτό να είναι εγγυημένο. Εάν υπάρχουν πολλά εκκρεμή bulk transfers, ο Host Controller θα προσπαθήσει να τα εξυπηρετήσει όλα με «δικαιοσύνη». Ο τρόπος με τον οποίο επιτυγχάνεται αυτό εξαρτάται από την υλοποίηση.

Η καθυστέρηση πραγματοποίησης μιας ογκώδους μεταφοράς κυμαίνεται σημαντικά. Εξαρτάται από τον αριθμό των συνδεδεμένων συσκευών στο USB και από τον αριθμό των αιτήσεων για bulk transfers σε διαφορετικά endpoints. Έτσι ένα IRP για ένα bulk endpoint μπορεί να εξυπηρετηθεί σε ένα frame μόνο ή αντίθετα σε πολλά και ίσως όχι διαδοχικά frames.

2.3 Πρωτόκολλο Διαύλου του USB

Σε αυτή την ενότητα περιγράφεται το Πρωτόκολλο USB, δηλαδή οι κανόνες βάσει των οποίων γίνεται η μεταφορά δεδομένων στο δίαυλο USB. Όπως θα αναλυθεί στο Κεφάλαιο 4, η εφαρμογή του Πρωτοκόλλου είναι ευθύνη του Hardware του συστήματος USB. Το λογισμικό οδήγησης του Host Controller πρέπει να τροφοδοτεί το υλικό με τις κατάλληλες πληροφορίες για την συμπλήρωση των πεδίων των πακέτων USB. Επίσης, όπως θα φανεί στα Κεφάλαια 3 και 5, σε ορισμένες υλοποιήσεις του υλικού του USB Host, το λογισμικό πρέπει να διασπά κάθε αίτηση μεταφοράς δεδομένων σε μικρότερες αιτήσεις, έτσι ώστε κάθε αίτηση να αντιστοιχεί σε μία μόνο συναλλαγή (*Transaction*) στον δίαυλο. Τέλος, το λογισμικό είναι υπεύθυνο για τήρηση του πρωτοκόλλου συγχρονισμού (*Toggle Sequencing*), μεταξύ του Host και των συσκευών USB, στην πλευρά του USB Host. Για τους παραπάνω λόγους κρίνεται απαραίτητη η μελέτη της μορφής των πακέτων USB, της διαδοχής των transactions που συνιστούν μια μεταφορά δεδομένων ανάλογα με τον τύπο μεταφοράς δεδομένων, και του πρωτοκόλλου συγχρονισμού μεταξύ του Host και των συσκευών USB. Η παρουσίαση του Πρωτοκόλλου γίνεται «από κάτω προς τα πάνω». Αρχικά παρουσιάζονται τα διάφορα πεδία και η μορφή των πακέτων USB και στη συνέχεια παρουσιάζονται τα βήματα των διαφόρων transactions, ανάλογα με το transfer type του transaction. Τέλος δίνονται πληροφορίες για τον συγχρονισμό και την αντιμετώπιση σφαλμάτων στο USB.

2.3.1 Οριοθέτηση των Πακέτων USB

Τα bits των διαφόρων πεδίων των πακέτων στέλνονται στο USB με σειρά από το LSb προς το MSb. Όλα τα πακέτα ξεκινούν με ένα πεδίο συγχρονισμού *(SYNC)* το οποίο σηματοδοτεί την αρχή του πακέτου. Η περιγραφή της κωδικοποίησης του πεδίου SYNC είναι εκτός του σκοπού της εργασίας. Μετά το SYNC ακολουθεί το επόμενο πεδίο του πακέτου. Το πεδίο SYNC δεν εμφανίζεται στα διαγράμματα πακέτων που ακολουθούν.

Η αρχή και το τέλος κάθε πακέτου ορίζονται σαφώς. Το Start Of Packet *(SOP)* αποτελεί μέρος του πεδίου SYNC. Το End Of Packet *(EOP)* ορίζεται σαφώς, αλλά η παρουσίαση της μορφής του ξεφεύγει από το σκοπό του παρόντος συγγράμματος **[1]**.

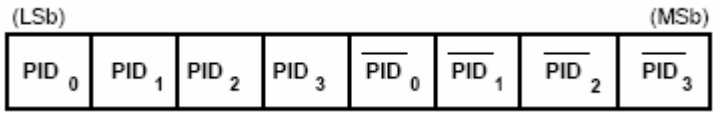
2.3.2 Πεδία των Πακέτων USB

Σε αυτή την ενότητα παρουσιάζεται η μορφή των πεδίων *(Fields)* των πακέτων Token, Data και Handshake. Τα bits των πεδίων υφίστανται κωδικοποίηση *(bit stuffing)*[*] από το hardware πριν σταλούν στο USB. Για λόγους σαφήνειας τα bits παρουσιάζονται σε μη κωδικοποιημένη μορφή.

* Τα bits, πριν από την εκπομπή τους στο δίαυλο, κωδικοποιούνται από τις συσκευές έτσι, ώστε να μην υπάρχει ακολουθία από περισσότερα από έξι συνεχόμενα 1. Η κωδικοποίηση αυτή ονομάζεται Bit Stuffing. Η ανίχνευση μιας ακολουθίας από επτά ή περισσότερα συνεχόμενα 1 προκαλεί Bit Stuff Error, με αποτέλεσμα το πακέτο που περιέχει το σφάλμα να θεωρείται λανθασμένο.

2.3.2.1 Πεδίο Packet Identifier

Το Packet Identifier (PID) ακολουθεί το SYNC σε κάθε πακέτο USB. Το PID αποτελείται από τέσσερα bits ακολουθούμενα από τα συμπληρώματά τους *(Check Bits)* για επαλήθευση. Το PID δηλώνει τον τύπο του πακέτου και επομένως τη μορφοποίηση του πακέτου και το είδος της ανίχνευσης λαθών για αυτό το πακέτο. Ένα PID Error συμβαίνει όταν τα Check Bits δεν είναι συμπληρώματα των αντίστοιχων PID bits.



Σχήμα 2.3.1: Πεδίο PID

Ο Host και όλες οι συσκευές αποκωδικοποιούν όλα τα PID που λαμβάνουν. Εάν προκύψει PID Error, ή η τιμή του πεδίου δεν αντιστοιχεί σε έγκυρη τιμή PID, το PID θεωρείται φθαρμένο και όλο το πακέτο αγνοείται. Αν ένα function λάβει ένα PID το οποίο αντιστοιχεί σε transaction ή σε κατεύθυνση που δεν υποστηρίζει, το function αγνοεί το πακέτο. Στον παρακάτω Πίνακα παρουσιάζονται οι έγκυρες κωδικοποιήσεις των PID.

Πίνακας 2.3.1: Κωδικοποίηση των PID

PID Type	PID Name	PID bits [3:0]	Description
Token	OUT	0001b	Host-to-function transaction
	IN	1001b	Function-to-Host transaction
	SOF	0101b	Start-of-frame και frame number
	SETUP	1101b	Host-to-function transaction για setup ενός control pipe
Data	DATA0	0011b	Άρτιο data packet PID
	DATA1	1011b	Περιττό data packet PID
Handshake	ACK	0010b	Ο δέκτης λαμβάνει πακέτο χωρίς λάθη
	NAK	1010b	Ο δέκτης δεν μπορεί να λάβει δεδομένα ή ο πομπός δεν μπορεί να στείλει δεδομένα
	STALL	1110b	Το endpoint είναι Halted, ή το control pipe request δεν υποστηρίζεται
Special	PRE	1100b	Low-Speed Preamble, επιτρέπει τη μεταφορά δεδομένων προς low-speed συσκευές

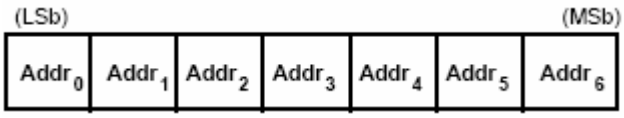
2.3.2.2 Πεδία Διευθύνσεων

Οι διευθύνσεις των endpoints των functions δίνονται από τον συνδυασμό των τιμών δύο πεδίων: Το Function Address Field και το Endpoint Field. Αν ο

συνδυασμός των πεδίων δεν δίνει κάποιο έγκυρο endpoint, το token θα αγνοηθεί. Το ίδιο θα συμβεί όταν το endpoint το οποίο δείχνουν τα πεδία δεν έχει αρχικοποιηθεί.

2.3.2.2.1 Πεδίο Address

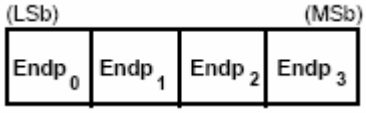
Το Address Field *(ADDR)* δίνει τη διεύθυνση του function που θα χρησιμοποιηθεί ως πομπός ή δέκτης του πακέτου δεδομένων. Το πεδίο αυτό περιέχεται σε IN, OUT και SETUP tokens και αποτελείται από 7 bits , δίνοντας 128 πιθανές διευθύνσεις.



Σχήμα 2.3.2: Address Field

2.3.2.2.2 Πεδίο Endpoint

Το Endpoint Field *(ENDP)* δίνει τον αριθμό του endpoint του function που συμμετέχει στο transaction. Οι αριθμοί των endpoints ορίζονται από το function. Το πεδίο αυτό περιέχεται σε IN, OUT και SETUP tokens και αποτελείται από 4 bits. Όλες οι συσκευές πρέπει να υποστηρίζουν ένα control pipe στο endpoint 0. Οι low-speed συσκευές υποστηρίζουν το πολύ τρια pipes ανά function και επομένως υποστηρίζουν το πολύ 3 endpoints. Οι full-speed functions μπορούν να έχουν μέχρι και 16 endpoints.



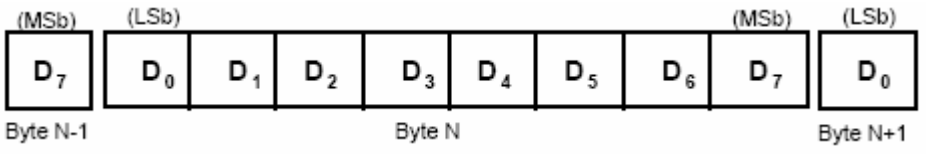
Σχήμα 2.3.3: Endpoint Field

2.3.2.3 Πεδίο Frame Number

Το Frame Number Field αποτελείται από 11 bits και η τιμή του αυξάνεται από τον Host σε κάθε frame. Όταν η τιμή του φτάσει τη μέγιστη τιμή 7FFh, το πεδίο μηδενίζεται στο επόμενο frame. Στέλνεται μόνο σε SOF tokens στην αρχή κάθε frame.

2.3.2.4 Πεδίο Data

Το μέγεθος του Data Field κυμαίνεται από 0 μέχρι 1023 bytes και αποτελείται από ακέραιο αριθμό bytes. Τα bits κάθε byte τοποθετούνται με το LSb πρώτο.



Σχήμα 2.3.4: Data Field

2.3.2.5 Πεδία Προστασίας CRC

Οι Πρόσθετοι Κυκλικοί Έλεγχοι (*Cyclic Redundancy Checks, CRCs)* αποτελούν επιπρόσθετα πεδία που παράγονται για την προστασία και τον έλεγχο της ορθότητας όλων των πεδίων, εκτός από τα PID, σε πακέτα token και data. Τα πεδία αυτά δημιουργούνται στον πομπό πριν την κωδικοποίησή (bit stuffing) τους. Στον δέκτη, τα πεδία αυτά αποκωδικοποιούνται μετά την αφαίρεση του bit stuffing. Τα CRC παρέχουν 100% προστασία από σφάλματα ενός ή δύο bit. Αν ένα CRC είναι λανθασμένο, θεωρείται ότι ένα ή περισσότερα από τα προστατευόμενα πεδία είναι φθαρμένα και αγνοούνται από τον δέκτη. Τα CRC βασίζονται στους πολυωνυμικούς κώδικες για ανίχνευση λαθών και παράγονται αυτόματα από το hardware. Η περεταίρω μελέτη τους είναι εκτός του θέματος του παρόντος συγγράμματος **[1]**.

2.3.2.5.1 CRC των Πακέτων Token

Ένα πεδίο CRC μήκους 5 bit παρέχεται για την προστασία των token και καλύπτει τα πεδία ADDR και ENDP των IN, SETUP και OUT tokens, ή το frame number field ενός SOF token.

2.3.2.5.2 CRC στα Πακέτα Data

Αυτό το πεδίο CRC μήκους 16 bits παρέχεται για την προστασία των data packets και καλύπτει το data field.

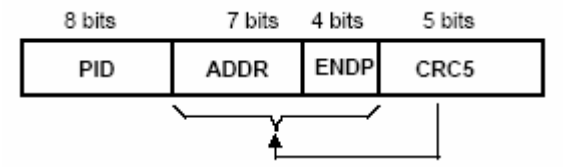
2.3.3 Μορφή των Πακέτων USB

Σε αυτή την ενότητα περιγράφεται η μορφή των πακέτων Token, Data και Handshake. Η σειρά των πεδίων στα σχήματα είναι αυτή με την οποία στέλνονται τα bits στο USB.

2.3.3.1 Πακέτα Token

Τα πακέτα Token αποτελούνται από ένα πεδίο PID, ακολουθούμενο από πεδία ADDR και ENDP. Οι τιμές του PID μπορεί να είναι IN, OUT ή SETUP. Για OUT ή SETUP PID *(Host-to-function transaction)* τα πεδία ADDR και ENDP δίνουν τη

διεύθυνση του endpoint το οποίο θα λάβει πακέτο Data που θα ακολουθήσει. Για IN PID *(Function-to-host transaction)*, δίνουν τη διεύθυνση του endpoint που θα στείλει το ένα πακέτο Data στον Host. Τα πακέτα Token μπορούν να σταλούν μόνο από τον Host.

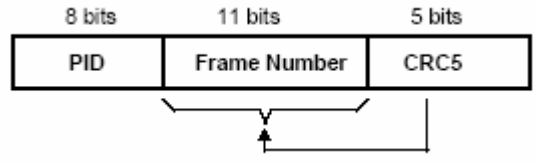


Σχήμα 2.3.5: Token Packet

Τέλος, τα πακέτα Token περιέχουν ένα πεδίο CRC μήκους 5 bits, για την προστασία των πεδίων διεύθυνσης του πακέτου.

2.3.3.2 Πακέτα Start-of-Frame

Ένα πακέτο SOF στέλνεται από τον Host στην αρχή ενός frame (ένα πακέτο κάθε 1.00 ms ± 0.0005 ms). Τα πακέτα SOF αποτελούνται από ένα PID ακολουθούμενο από ένα πεδίο Frame Number.

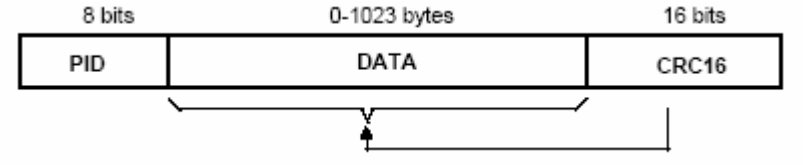


Σχήμα 2.3.6: SOF Packet

Το transaction που σηματοδοτεί ένα SOF, αποτελείται μόνο από αυτό το πακέτο. Δεν ακολουθεί άλλο πακέτο ως μέρος του transaction. Όλες οι full-speed συσκευές, συμπεριλαμβανομένων των Hubs, λαμβάνουν το SOF. Οι συσκευές δεν στέλνουν πακέτο στον Host ως απάντηση. Για το λόγο αυτό η παραλαβή των SOF δεν είναι εγγυημένη. Το πεδίο CRC εφαρμόζεται στο Frame Number.

2.3.3.3 Πακέτα Data

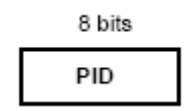
Τα πακέτα Data αποτελούνται από ένα PID, ένα πεδίο data αποτελούμενο από 0 έως 1023 bytes και ένα πεδίο CRC 16 bits, το οποίο εφαρμόζεται στο πεδίο data. Το πεδίο PID μπορεί να έχει τιμές DATA0 ή DATA1 για την υποστήριξη συγχρονισμού στην μεταφορά των δεδομένων (Data Toggle Synchronization).



Σχήμα 2.3.7: Data Packet

2.3.3.4 Πακέτα Handshake

Τα πακέτα Handshake αποτελούνται μόνο από ένα πεδίο PID. Χρησιμοποιούνται για να αναφέρουν το status ενός transaction και επιστρέφουν τιμές που υποδεικνύουν επιτυχή παραλαβή των δεδομένων, αποδοχή ή απόρριψη εντολής, έλεγχο ροής *(flow control)* και καταστάσεις Halt. Τα Handshake επιστρέφονται μόνο σε transactions που έχουν έλεγχο ροής δεδομένων. Επιστρέφονται κατά την φάση Handshake ενός transaction. Μπορούν να επιστραφούν και στην φάση Data αντί για δεδομένα.



Σχήμα 2.3.8: Handshake Packet

Υπάρχουν τρία είδη πακέτων Handshake:

- **ACK**. Δηλώνει ότι το πακέτο Data ελήφθη χωρίς σφάλματα bit stuff ή CRC και το PID ήταν σωστό. Στέλνεται όταν τα sequence bits ταιριάζουν και ο δέκτης μπορεί να δεχθεί τα δεδομένα ή όταν δεν είναι ίδια και πρέπει να γίνει συγχρονισμός μεταξύ πομπού και δέκτη (Ενότητα 2.5). Στέλνεται από τον Host σε IN transactions και από ένα function σε OUT transactions και μόνο σε transactions σε οποία στέλνονται δεδομένα και αναμένεται ένα Handshake.

- **NAK**. Στέλνεται μόνο από functions και δηλώνει ότι ένα function δεν μπορεί να λάβει δεδομένα (OUT) ή ότι δεν έχει δεδομένα για αποστολή (IN), αλλά τελικά θα μπορέσει χωρίς παρέμβαση του Host. Σε ένα IN transaction στέλνεται κατά τη φάση Data και σε ένα OUT κατά τη φάση Handshake. Στέλνεται για λόγους ελέγχου ροής δεδομένων.

- **STALL**. Στέλνεται μόνο από functions σαν απάντηση σε ένα IN token ή κατά τη φάση Data ενός OUT transaction. Δηλώνει αδυναμία αποστολής ή λήψης δεδομένων, ή ότι μια αίτηση για control pipe δεν υποστηρίζεται. Υπάρχουν δύο περιπτώσεις STALL. Το "Functional Stall" συμβαίνει όταν το endpoint είναι Halt ως αποτέλεσμα ρητής εντολής του Host. Το "Protocol Stall" συμβαίνει σε control pipes λόγω λανθασμένη εντολής, επιστρέφεται στο Data ή Status stage ενός control transfer και τερματίζεται στην αρχή του επόμενου control transfer.

2.3.3.5 Απαντήσεις με Handshake

Στους παρακάτω πίνακες καταγράφονται τα πιθανά Handshakes ή πακέτα που επιστρέφονται από ένα function ή από τον Host, ανάλογα με το transaction (IN ή OUT) και τις συνθήκες.

Πίνακας 2.3.2 Απάντηση ενός Function σε ένα IN Token

Το λαμβανόμενο Token είναι Corrupted	Το Endpoint είναι Halt	Το Function μπορεί να στείλει δεδομένα	Απάντηση
ΝΑΙ	Χ	Χ	-
ΟΧΙ	ΝΑΙ	Χ	STALL
ΟΧΙ	ΟΧΙ	ΟΧΙ	NAK
ΟΧΙ	ΟΧΙ	ΝΑΙ	Αποστολή πακέτου Data

Πίνακας 2.3.3 Απάντηση του Host σε ένα πακέτο Data (IN transaction)

Το λαμβανόμενο Data Packet είναι Corrupted	Ο Host μπορεί να λάβει δεδομένα	Handshake επιστρεφόμενο από τον Host
ΝΑΙ	Χ	Απόρριψη δεδομένων, κανένα Handshake
ΟΧΙ	ΟΧΙ	Απόρριψη δεδομένων, κανένα Handshake
ΟΧΙ	ΝΑΙ	Αποδοχή πακέτου, αποστολή ACK

Πίνακας 2.3.4 Απάντηση ενός Function σε ένα πακέτο Data (OUT transaction)

Το λαμβανόμενο Token είναι Corrupted	Το Endpoint είναι Halt	Τα Sequence Bits ταιριάζουν	Το Function μπορεί να λάβει δεδομένα	Handshake επιστρεφόμενο από τον Function
ΝΑΙ	Χ	Χ	Χ	-
ΟΧΙ	ΝΑΙ	Χ	Χ	STALL
ΟΧΙ	ΟΧΙ	ΟΧΙ	Χ	ACK
ΟΧΙ	ΟΧΙ	ΝΑΙ	ΝΑΙ	ACK
ΟΧΙ	ΟΧΙ	ΝΑΙ	ΟΧΙ	NAK

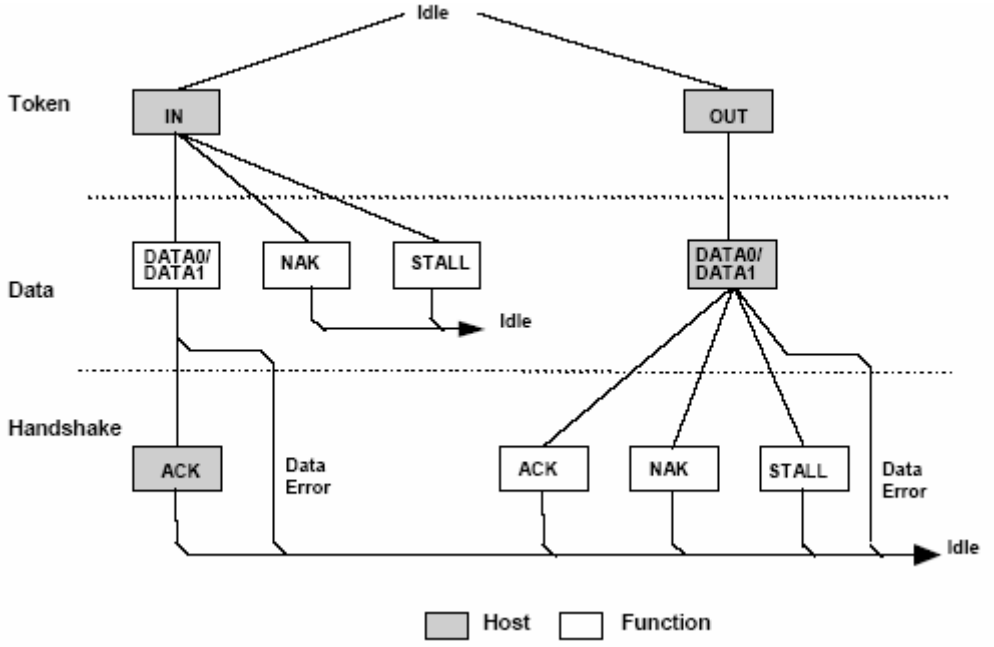
Ένα SETUP transaction επιτρέπει στον Host να συγχρονίσει τα bits συγχρονισμού του function με τα δικά του. Με τη λήψη του SETUP token η συσκευή πρέπει να δεχθεί τα δεδομένα που ακολουθούν το token. Δεν επιτρέπεται η απάντηση του function με NAK ή STALL. Αν το endpoint που δέχεται το token είναι non-control απλώς αγνοεί το token.

2.3.4 Μορφή των Transactions

Η μορφή ενός transaction εξαρτάται από τον τύπο του endpoint που συμμετέχει σε αυτό. Υπάρχουν τέσσερα είδη endpoint: bulk, control, interrupt και isochronous. Τα τρία πρώτα διαθέτουν μηχανισμούς ανίχνευσης λαθών.

2.3.4.1 Συναλλαγές Bulk (Bulk Transactions)

Τα Bulk transactions πραγματοποιούνται σε τρεις φάσεις και αποτελούνται από πακέτα token, data και handshake. Υπό ορισμένες συνθήκες η φάση data αντικαθιστάται από ένα handshake, με αποτέλεσμα το transaction να ολοκληρώνεται σε δύο φάσεις χωρίς μεταφορά δεδομένων.

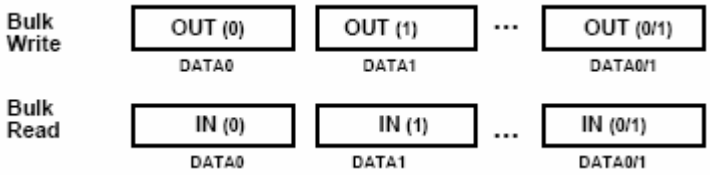


Σχήμα 2.3.9: Μορφή των Bulk Transactions

‘Οπως φαίνεται στο παραπάνω σχήμα, για να πραγματοποιηθεί ένα IN bulk transaction ο Host στέλνει ένα IN token. Η συσκευή στέλνει ένα πακέτο data, ή εάν δεν μπορεί να το στείλει, στέλνει ένα NAK ή STALL handshake (ανάλογα με την αιτία) με αποτέλεσμα τον τερματισμό του transaction. Αν το πακέτο data που λαμβάνει ο Host δεν περιέχει σφάλματα, ο Host στέλνει ένα ACK handshake, αλλιώς δεν απαντά.

Για να πραγματοποιηθεί ένα OUT bulk transaction ο Host στέλνει ένα OUT token ακολουθούμενο από ένα πακέτο data. Αν το πακέτο data που λαμβάνει η συσκευή δεν περιέχει σφάλματα, στέλνει ένα ACK handshake δηλώνοντας ότι μπορεί να σταλεί το επόμενο πακέτο data, αλλιώς δεν απαντά. Σε περίπτωση που δεν μπορεί να λάβει δεδομένα στέλνει NAK ή STALL ανάλογα με την αιτία.

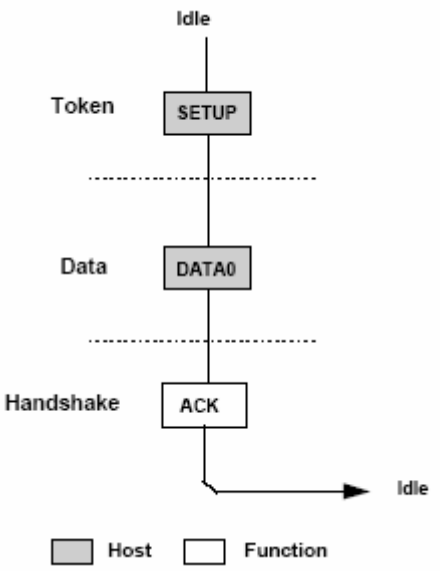
Στο επόμενο σχήμα φαίνεται ο συγχρονισμός των πακέτων data σε ένα bulk endpoint. Ο Host πάντα αρχικοποιεί το πρώτο transaction στο endpoint με PID DATA0 ρυθμίζοντας κατάλληλα το endpoint *(Configuration Event)*. Τότε το sequence bit του endpoint τίθεται ίσο με DATA0. Στη συνέχεια το transaction χρησιμοποιεί DATA1 PID. Από εκεί και πέρα σε κάθε μεταφορά δεδομένων το PID του πακέτου θα αλλάζει *(toggle)* σε σχέση με την προηγούμενη. Ο πομπός αλλάζει sequence bit με τη λήψη ACK και ο δέκτης με τη λήψη πακέτου data χωρίς σφάλματα *(Toggle Sequencing)*.



Σχήμα 2.3.10: Συγχρονισμός στα Bulk Transactions

2.3.4.2 Μεταφορές Ελέγχου (Control Tranfers)

Τα Control Transfers πραγματοποιούνται σε τουλάχιστον δύο στάδια: Setup και Status. Μπορεί να υπάρχει και ένα στάδιο Data ανάμεσα στα δύο αυτά στάδια. Στο στάδιο Setup πραγματοποιείται ένα SETUP transaction για τη μεταφορά δεδομένων προς το endpoint. Το token του transaction έχει PID SETUP και χρησιμοποιείται μόνο PID DATA0 για το πακέτο data. Το endpoint που λαμβάνει τα δεδομένα στέλνει ένα ACK αν τα δεδομένα δεν έχουν σφάλματα, αλλιώς δεν στέλνει Handshake.

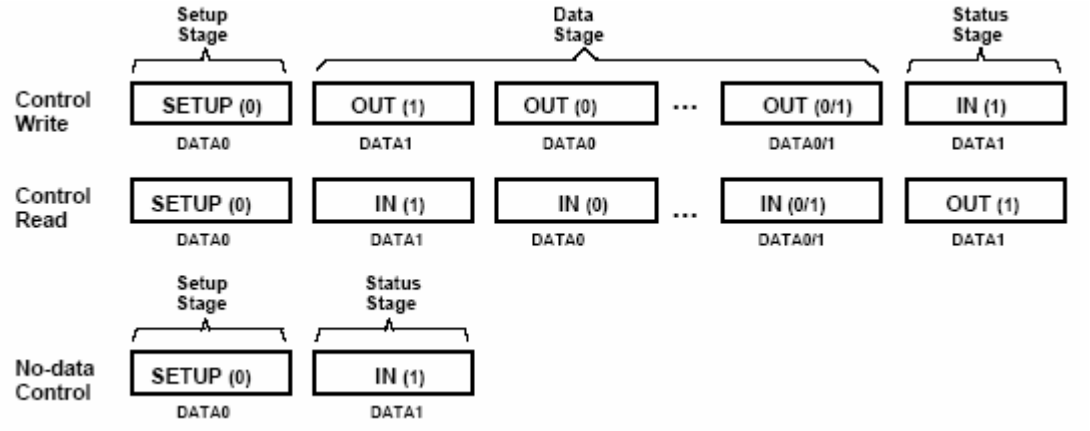


Σχήμα 2.3.11: Μορφή των Control SETUP Transactions

Το στάδιο Data, αν υπάρχει, αποτελείται από ένα ή περισσότερα IN ή OUT transactions (μια κατεύθυνση μόνο) και ακολουθεί τους ίδους κανόνες πρωτοκόλλου με τα bulk transactions. Το μέγεθος των δεδομένων καθορίζεται κατά το στάδιο Setup

και τα δεδομένα μεταφέρονται σε πολλά πακέτα, αν το μέγεθός τους ξεπερναέι το μέγεθος του ενός πακέτου.

Στο στάδιο Status παρατηρείται μεταβολή στην κατεύθυνση της μεταφοράς των δεδομένων, αποτελείται από ένα transaction, και χρησιμοποιεί πάντα PID DATA1. Αν για παράδειγμα το προηγούμενο στάδιο έχει κατεύθυνση OUT, το στάδιο Status έχει κατεύθυνση IN. Στο παρακάτω σχήμα παρουσιάζεται ο συγχρονισμός των Control Transfers και η σειρά των transactions. Αν ένα control endpoint επιστρέψει STALL στο στάδιο data ή status, το ίδιο handshake πρέπει να επιστρέφεται μέχρι να λάβει ένα SETUP PID.



Σχήμα 2.3.12: Συγχρονισμός στα Control Transactions

Στο στάδιο Status το function αναφέρει στον Host το αποτέλεσμα των προηγούμενων σταδίων *(Status Reporting)*. Στον παρακάτω πίνακα συνοψίζονται οι απαντήσεις που δίνονται κατά το στάδιο status *(Status Responses)*.

Πίνακας 2.3.5: Απαντήσεις στο Στάδιο Status

Απάντηση Status	**Control Write (data phase)**	**Control Read (handshake phase)**
Η εντολή ολοκληρώθηκε	Πακέτο data μηδενικού μήκους	ACK
Η εντολή απέτυχε να ολοκληρωθεί	STALL	STALL
Το function επεξεργάζεται ακόμα την εντολή	NAK	NAK

Στις Control Read transfers το status επιστρέφεται κατά τη φάση Handshake του σταδίου Status, αφού ο Host έχει στέλει ένα πακέτο data μηδενικού μήκους κατά τη φάση data. Στις Control Write transfers το status επιστρέφεται κατά τη φάση data του σταδίου Status. Στην περίπτωση ολοκλήρωσης της εντολής σε Control Write το function αναμένει από τον Host να απαντήσει με ένα ACK. Αν δεν το λάβει θα συνεχίσει να στέλνει ένα πακέτο data μηδενικού μήκους όσο ο host στέλνει IN tokens.

Σε κάθε περίπτωση η απάντηση με NAK σημαίνει ότι ο host πρέπει να παραμείνει στο στάδιο status. Η απάντηση με STALL αντιστοιχεί σε protocol STALL και σημαίνει ότι η εντολή προς το function ή οι παράμετροί της δεν έγιναν κατανοητά. Το function θα στέλνει STALL σε κάθε αίτηση από τον Host μέχρι να λάβει ένα SETUP token. Τα functional Stall δεν πρέπει να υποστηρίζονται από control endpoints.

Αν σε ένα στάδιο Data ενός Control Pipe στέλνονται περισσότερα δεδομένα ή ζητείται να σταλούν περισσότερα δεδομένα από αυτά που δηλώθηκαν κατά το στάδιο Setup, επιστρέφεται STALL κατά τη φάση αυτή και δεν ακολουθεί στάδιο Status. Ωστώσο, η φάση data μπορεί να είναι μεταβλητού μήκους στην οποία ο Host ζητάει περισσότερα δεδομένα από αυτά που περιέχονται στη συγκεκριμένη δομή δεδομένων. Σε αυτή την περίπτωση το function δηλώνει το τέλος του σταδίου data με την αποστολή πακέτου data μήκους μικρότερου από το MaxPacketSize του pipe, ή μηδενικού μήκους αν ο όγκος των δεδομένων της δομής είναι πολλαπλάσιο του MaxPacketSize.

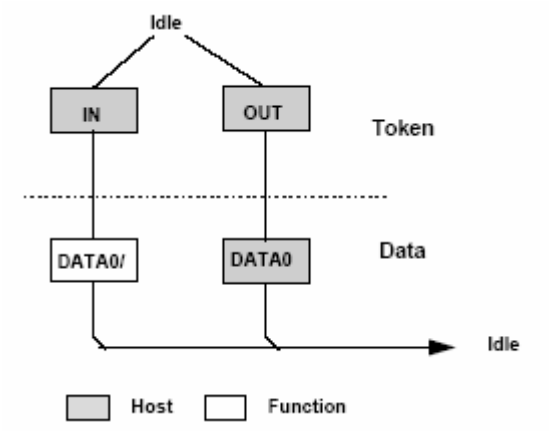
2.3.4.3 Συναλλαγές Interrupt (Interrupt Transactions)

Τα Interrupt transactions υλοποιούνται με όμοιο τρόπο με τα bulk. Το σχήμα που τα περιγράφει είναι ίδιο με το σχήμα 2.3.9. Σε ένα IN transaction, το endpoint στέλνει NAK αν δεν έχει πληροφορίες να στείλει. Αν είναι σε κατάσταση Halt στέλνει STALL. Αν εκκρεμεί interrupt (το endpoint έχει δεδομένα για εκπομπή) στέλνει ένα πακέτο data. Ο Host απαντάει ανάλογα με ένα handshake.

Όταν το endpoint στέλνει interrupt data, το πρωτόκολλο που περιγράφτηκε στα bulk transactions για τον συγχρονισμό των πακέτων data *(Toggle Sequencing)* πρέπει να ακολουθείται. Αυτό επιτρέπει στο function να γνωρίζει ότι τα δεδομένα ελήφθησαν από τον Host και το interrupt condition (η κατάσταση που προκάλεσε την αποστολή δεδομένων) να σβηστεί. Έτσι το endpoint θα συνεχίσει να στέλνει δεδομένα στον Host μέχρι να επιβεβαιωθεί ότι παραδόθηκαν. Με αυτό τον τρόπο δε χρειάζεται να στέλνονται δεδομένα όποτε το function γίνεται polled ή μέχρι να σβηστεί το interrupt condition από το λογισμικό.

2.3.4.4 Συναλλαγές Isochronous (Isochronous Transactions)

Τα isochronous transactions έχουν μια φάση token και μια φάση data, αλλά όχι φάση handshake. Έτσι δεν έχουν δυνατότητα επανάληψης της μετάδοσης σε περίπτωση σφάλματος. Ένα device ή ο Host πρέπει να δέχεται πακέτα data με PID DATA0 ή DATA1. Όμως πρέπει να στέλνουν πακέτα με PID DATA0. Τα isochronous transactions δεν υποστηρίζουν toggle sequencing..



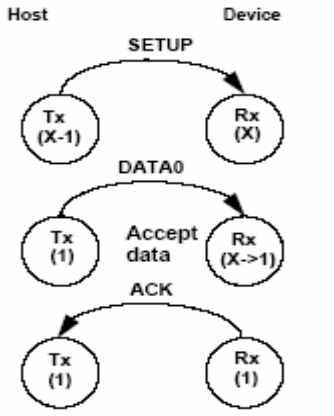
Σχήμα 2.3.13: Μορφή των Isochronous Transactions

2.3.5 Συγχρονισμός Toggle Sequencing

Το USB παρέχει ένα μηχανισμό για να εγγυηθεί τον συγχρονισμό των δεδομένων *(Data Sequence Synchronization)* μεταξύ πομπού και δέκτη σε πολλαπλά transactions. Έτσι εγγυάται ότι η φάση handshake του transaction ερμηνεύτηκε σωστά και από τις δύο πλευρές. Ο συγχρονισμός επιτυγχάνεται μέσω των PID DATA0 και DATA1 και τη χρησιμοποίηση ξεχωριστών data toggle sequence bits στον πομπό και τον δέκτη. Τα sequence bits του πομπού και του δέκτη συγχρονίζονται στην αρχή του transaction. Ο μηχανισμός συγχρονισμού εξαρτάται από τον τύπο του transaction (δεν υποστηρίζεται σε isochronous).

2.3.5.1 Αρχικοποίηση με Κουπονί SETUP (SETUP Token)

Τα Control Transfers χρησιμοποιούν το SETUP token για την αρχικοποίηση των sequence bits στον host και το function, όπως φαίνεται στο παρακάτω σχήμα. Το function πρέπει να δεχθεί τα δεδομένα και να επιστρέψει ACK. Στη συνέχεια θέτει 1 το sequence bit του, έτσι ώστε στο τέλος του SETUP transaction τα sequence bits και στις δύο πλευρές να είναι 1.

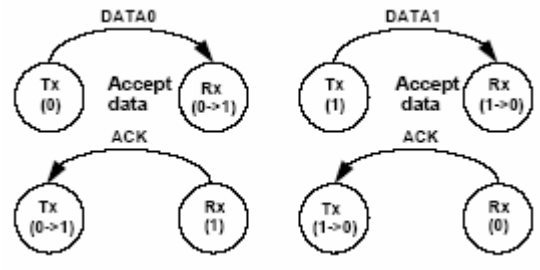


Σχήμα 2.3.14: Αρχικοποίηση με SETUP

2.3.5.2 Επιτυχημένες Συναλλαγές Δεδομένων

Το παρακάτω σχήμα περιγράφει την περίπτωση δύο επιτυχών Data transactions. Τα sequence bits του δέκτη αλλάζουν *(toggle)* όταν ο δέκτης μπορεί να λάβει δεδομένα και λαμβάνει ένα πακέτο data χωρίς σφάλματα και με PID που ταιριάζει με την τρέχουσα τιμή του sequence bit του. Τα sequence bits του πομπού αλλάζουν όταν ο πομπός λαμβάνει ένα ACK handshake.

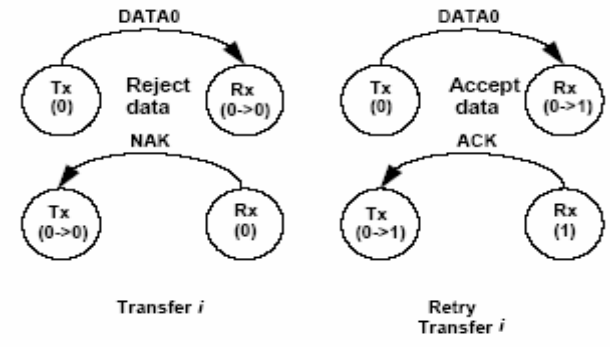
Σε κάθε transaction ο δέκτης συγκρίνει το sequence bit του πομπού, το οποίο κωδικοποιείται στο PID του πακέτου data ως DATA0 ή DATA1, με το δικό του sequence bit. Αν δεν μπορεί να δεχθεί τα δεδομένα, στέλνει NAK και τα sequence bits και στις δύο πλευρές δεν αλλάζουν. Αν μπορεί να δεχθεί τα δεδομένα και το sequence bit του ταιριάζει με το PID του πακέτου, δέχεται τα δεδομένα και αλλάζει το sequence bit. Στα transactions που πραγματοποιούνται σε δύο φάσεις και δεν υπάρχει πακέτο data, τα sequence bits δεν αλλάζουν στον πομπό και τον δέκτη.



Σχήμα 2.3.15: Συνεχή Transactions

2.3.5.3 Αποτυχημένες Συναλλαγές Δεδομένων

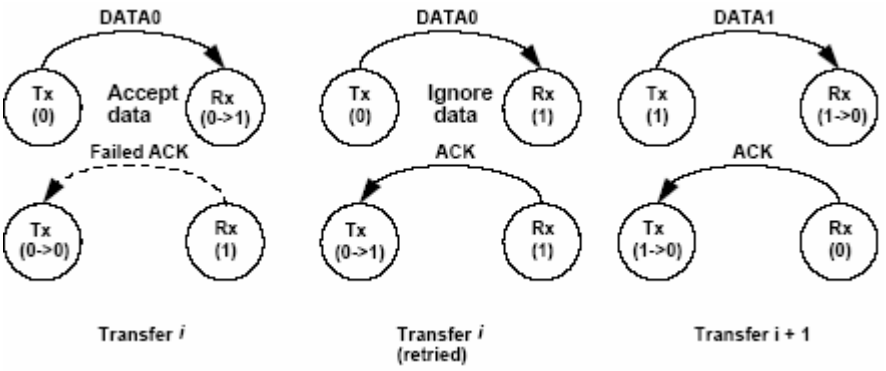
Αν ο δέκτης δεν μπορεί να δεχθεί τα δεδομένα ή το πακέτο data έχει σφάλματα, ο δέκτης θα στείλει NAK ή STALL, ή θα προκύψει timeout, ανάλογα με την αιτία, και ο δέκτης δεν θα αλλάξει το sequence bit του. Στο παρακάτω σχήμα φαίνεται η περίπτωση που επιστρέφεται NAK και το transaction πραγματοποιείται ξανά *(Retry)*. Η ίδια διαδικασία ακολουθείται για κάθε αιτία που προκαλεί σφάλμα. Ο πομπός αφού δεν δέχεται ACK δεν θα αλλάξει το sequence bit του. Έτσι τα sequence bits των δύο πλευρών παραμένουν συγχρονισμένα. Το transaction πραγματοποιείται ξανά και, αν είναι επιτυχές, τα sequence bits των δύο πλευρών θα αλλάξουν.



Σχήμα 2.3.16: Αποτυχημένο Transaction και Επανάληψη

2.3.5.4 Αποτυχημένη Χειραψία ACK

Ο πομπός, σε αντίθεση με τον δέκτη, είναι ο μόνος που γνωρίζει αν τελικά το transaction ήταν επιτυχές, με τη λήψη του ACK handshake. Η αποτυχία του ACK οδηγεί σε ένα προσωρινό χάσιμο του συγχρονισμού μεταξύ πομπού και δέκτη, όπως φαίνεται στο επόμενο σχήμα.



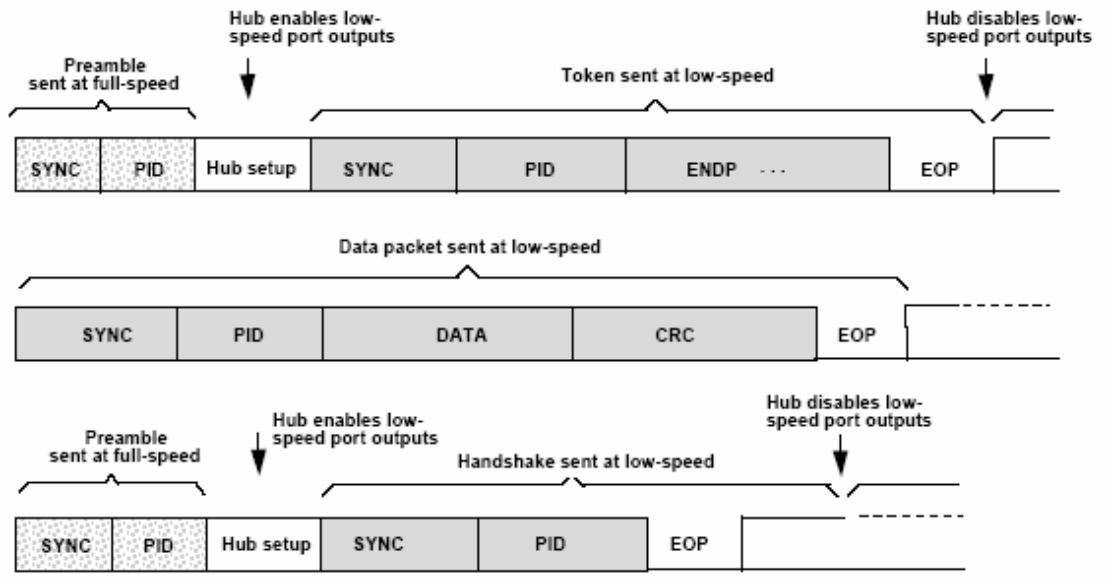
Σχήμα 2.3.17: Αποτυχημένο ACK Handshake και Επανάληψη

Στο τέλος του transaction *i*, δημιουργείται ένα προσωρινό χάσιμο του συγχρονισμού μεταξύ πομπού και δέκτη, γιατί τα sequence bits των δύο πλευρών δεν ταιριάζουν. Ο δέκτης έχει λάβει έγκυρα δεδομένα αλλά ο πομπός δεν το γνωρίζει. Στο επόμενο transaction ο πομπός θα ξαναστείλει το ίδιο πακέτο data με PID DATA0. Το sequence bit του δέκτη δεν ταιριάζει με το data PID και έτσι ο δέκτης γνωρίζει ότι έχει λάβει το πακέτο αυτό προηγουμένως. Επομένως δεν δέχεται το πακέτο data, δεν αλλάζει το sequence bit του, και στέλνει ACK. Ο πομπός ερμηνεύει το ACK ως επιτυχία του transaction, και αλλάζει το sequence bit του. Στην αρχή του transaction $i+1$ ο πομπός και ο δέκτης θα είναι συγχρονισμένοι.

Ο πομπός πρέπει να εξασφαλίζει ότι κάθε πακέτο που ξαναστέλνεται πρέπει να είναι ίδιο με το πακέτο που στάλθηκε στο αρχικό transaction. Αν αυτό δεν είναι δυνατόν, λόγω προβλημάτων όπως buffer underrun, ο πομπός πρέπει να εγκαταλείψει το transaction δημιουργώντας ένα bit stuff error. Ο δέκτης θα ανιχνεύσει εγγυημένα το σφάλμα. Ο πομπός δεν πρέπει να εξαναγκάσει την εμφάνιση σφάλματος με ένα λανθασμένο CRC, γιατί ο συνδυασμός ενός λανθασμένου πακέτου με ένα λανθασμένο CRC μπορεί να ερμηνευτεί από τον δέκτη ως έγκυρο πακέτο.

2.3.5.5 Συναλλαγές Χαμηλής Ταχύτητας (Low Speed Transactions)

Όπως έχει ήδη αναφερθεί, το USB υποστηρίζει full-speed και low-speed σηματοδοσία. Τα hubs εμποδίζουν την downstream κίνηση σε όλα τα ports στα οποία συνδέονται low-speed συσκευές, κατά τη διάρκεια downstream full-speed σηματοδοσίας. Αυτό είναι απαραίτητο, ώστε να μην υπάρχει πιθανότητα να ερνηνευτεί λανθασμένα από μια low-speed συσκευή ένα full-speed πακέτο. Το παρακάτω σχήμα δείχνει ένα IN low-speed transaction στο οποίο ο host στέλνει ένα token και ένα handshake και λαμβάνει ένα πακέτο.



Σχήμα 2.3.18: Low-Speed Transaction

Όλα τα downstream πακέτα που στέλνονται σε low-speed συσκευές απαιτούν ένα «προοίμιο» *(preamble)*. Το preamble αποτελείται από ένα SYNC ακολουθούμενο από ένα PID PRE, που στέλνονται σε full-speed. Τα hubs πρέπει να αναγνωρίζουν τα PRE, ενώ όλες οι άλλες συσκευές μπορούν να τα αγνοούν. Μετά το τέλος του preamble ο host πρέπει να περιμένει για χρόνο τουλάχιστον τεσσάρων full-speed bits, στον οποίο τα hubs πρέπει να ολοκληρώσουν την διαδικασία την ενεργοποίησης του επαναλήπτη στα ports που συνδέονται low-speed συσκευές. Σε αυτό τον χρόνο τα hubs πρέπει να θέσουν όλα τα ports σε ανενεργή κατάσταση *(Idle)* και να είναι έτοιμα για low-speed σηματοδοσία στα low-speed ports. Οι κανόνες για τη σύνδεση low-speed συσκευών είναι οι εξής:

1. Οι low-speed συσκευές αναγνωρίζονται κατά τη σύνδεσή τους, και τα ports στα οποία συνδέονται αναγνωρίζονται ως low-speed.

2. Όλα τα downstream low-speed πακέτα πρέπει να έπονται ενός full-speed preamble, το οποίο θέτει σε λειτουργία τους buffers εξόδου στα low-speed ports των hubs.

3. Τα buffers εξόδου στα low-speed ports των hubs απενεργοποιούνται με τη λήψη ενός EOP και δεν ενεργοποιούνται αν δεν ανιχνεύσουν ένα preamble PID.

4. Η upstream συνδεσιμότητα δεν επηρρεάζεται από το αν ένα port είναι low-speed ή full-speed.

Η upstream low-speed σηματοδοσία ξεκινά με ένα low-speed SYNC, ακολουθούμενο από το υπόλοιπο πακέτο. Όταν αναγνωριστεί το EOP, τα hubs απενεργοποιούν τα low-speed ports προς την downstream κατεύθυνση. Τα ports παραμένουν ενεργά προς την upstream κατεύθυνση.

Τα low-speed και full-speed transactions έχουν πολλά κοινά χαρακηριστικά πρωτοκόλλου. Όμως η low-speed σηματοδοσία έχει τους εξής περιορισμούς:

- Το data payload περιορίζεται σε μέγιστο 8 bytes.
- Επιτρέπονται μόνο interrupt και control transfers.
- Το πακέτο SOF δεν λαμβάνεται από low-speed συσκευές.

2.3.6 Ανίχνευση Λαθών και Αποκατάσταση

Το USB παρέχει τρεις μηχανισμούς ανίχνευσης λαθών σε πακέτα: Παραβιάσεις του Bit Stuffing, Check Bits των PID, και CRCs. Οι μηχανισμοί αυτοί έχουν ήδη περιγραφτεί. Κάθε πακέτο που λαμβάνεται με σφάλματα αγνοείται από τον δέκτη. Ο παρακάτω πίνακας συνοψίζει τους μηχανισμούς ανίχνευσης σφαλμάτων, τους τύπους των πακέτων που εφαρμόζονται, και τις κατάλληλες αντιδράσεις του δέκτη.

Πίνακας 2.3.6: Τύποι Σφαλμάτων των Πακέτων USB

Πεδίο	Σφάλμα	Αντίδραση
PID	PID Check, Bit Stuff	Το πακέτο αγνοείται
Address	Bit Stuff, Address CRC	Το token αγνοείται
Frame Number	Bit Stuff, Frame Number CRC	Το πεδίο Frame Number αγνοείται
Data	Bit Stuff, Data CRC	Τα δεδομένα αγνοούνται

Εκτός από τα παραπάνω σφάλματα που οφείλονται σε λανθασμένα πεδία των πακέτων USB, παρατηρούνται τα εξής σφάλματα:

Timeout: Σε ένα transaction οι συσκευές που συμμετέχουν σε αυτό, ανταλλάσουν πακέτα ή handshakes μεταξύ τους. Μια συσκευή μπορεί να περιμένει για περιορισμένο χρόνο την απάντηση της άλλης συσκευής. Αν ο χρόνος αυτός εκπνεύσει χωρίς ληφθεί απάντηση, η μετάδοση θεωρείται αποτυχημένη και προκαλείται σφάλμα Timeout. Ο χρόνος αναμονής για τον Host είναι τουλάχιστον διάρκειας 18 bits, ενώ για τις υπόλοιπες συσκευές κυμαίνεται μεταξύ της διάρκειας 16 και 18 bits.

Babble: Ο δίαυλος πρέπει να βρίσκεται σε ανενεργή κατάσταση *(Idle)* στο τέλος του frame. Εάν η μεταφορά ενός πακέτου δεν έχει ολοκληρωθεί προς το τέλος του frame, ο δίαυλος θα παρουσιάζει δραστηριότητα, και θα προκληθεί σφάλμα Babble. Τα hubs επενεργοποιούν τα ports που έχουν δραστηριότητα κοντά στο τέλος του frame (πριν το σημείο EOF2), και αναφέρουν στον Host την ύπαρξη σφάλματος Babble. Το πακέτο που προκάλεσε το Babble θα έχει σφάλματα και θα αγνοηθεί από τον δέκτη. Ένα Babble μπορεί να προκληθεί λανθασμένα εάν δεν ανιχνευθεί επιτυχώς το EOP του τελευταίου πακέτου του frame, το οποίο κανονικά δεν θα ξεπερνούσε τον χρόνο του frame. Τότε θα θεωρηθεί ότι το πακέτο δεν έχει ολοκληρωθεί στο τέλος του frame και θα προκληθεί σφάλμα.

ΚΕΦΑΛΑΙΟ 3

Επικοινωνία Υλικού και Λογισμικού του USB

Στο κεφάλαιο αυτό περιγράφεται το υλικό και το λογισμικό του Host ενός συστήματος USB. Περιγράφεται ο ρόλος κάθε στρώματος, υλικού ή λογισμικού, του Host και ο τρόπος με τον οποίο επιτυγχάνεται η επικοινωνία μεταξύ γειτονικών στρωμάτων. Όπως έχει ήδη αναφερθεί στο Κεφάλαιο 2, η στρωματική δομή του Host επιτρέπει σε κάθε στρώμα να είναι ανεξάρτητο από τις λεπτομέρειες της υλοποίησης των υπόλοιπων στρωμάτων. Έτσι, κάθε στρώμα του Host εκτελεί διαφορετικές λειτουργίες σε συγχρονισμό με τα υπόλοιπα στρώματα, με σκοπό τη διαχείριση των συσκευών που είναι συνδεδεμένες στο USB και την πραγματοποίηση των απαιτούμενων μεταφορών δεδομένων.

Έμφαση δίνεται στον τρόπο με τον οποίο επιτυγχάνεται η επικοινωνία του Host Controller *(HC)*, δηλαδή του υλικού του USB Host, με το λογισμικό του συστήματος USB μέσω του Host Controller Driver *(HCD)*. Η υλοποίηση του HCD εξαρτάται από την υλοποίηση του HC, και τη διαπροσωπεία *(Interface)* της συγκεκριμένης υλοποίησης που παρέχεται προς το HCD. Συγκεκριμένα, υπάρχουν δύο τυποποιημένα interfaces του HC συμβατά με το πρωτόκολλο USB 1.1: Το Universal Host Controller Interface *(UHCI)* **[2]** και το Open Host Controller Interface *(OpenHCI)* **[3]**. Στην τελευταία ενότητα του κεφαλαίου δίνονται γενικές πληροφορίες για αυτά τα δύο interfaces. Στο Κεφάλαιο 4 και τα Παραρτήματα Α και Β περιγράφεται λεπτομερώς το UHCI, για το οποίο η συγγραφή ενός HCD που να το οδηγεί είναι το αντικείμενο της παρούσας εργασίας.

3.1 Υλικό και Λογισμικό του USB Host

Όπως έχει ήδη περιγραφεί στο κεφάλαιο 2, ο Host ενός συστήματος USB αποτελείται από τα εξής στρώματα λογισμικού και υλικού (από πάνω προς τα κάτω):

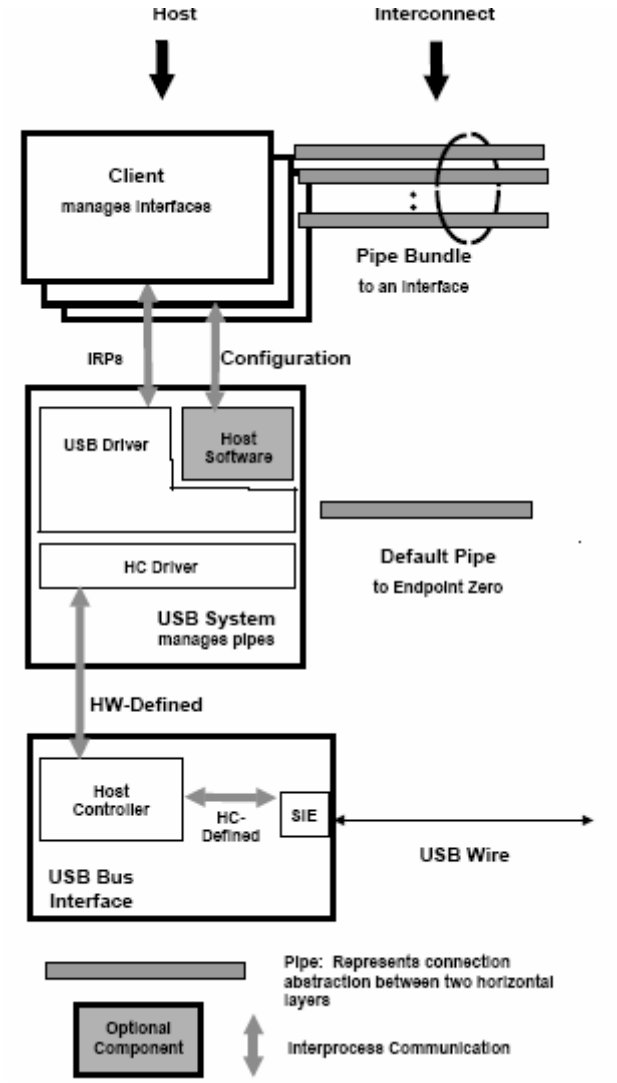
- Λογισμικό Συσκευής *(Client Software)*
- Λογισμικό Συστήματος USB *(USB System Software)*
- Ελεγκτής USB του Host *(USB Host Controller, HC)*

Η επικοινωνία μεταξύ δύο γειτονικών στρωμάτων του Host επιτυγχάνεται μέσω δύο διαπροσωπειών *(Interfaces)* λογισμικού που παρεμβάλλονται μεταξύ των στρωμάτων. Οι διαπροσωπείες αυτές είναι (από πάνω προς τα κάτω):

- Λογισμικό Οδήγησης του USB *(USB Driver, USBD)*: Το interface λογισμικού μεταξύ του USB System Software και του Client Software.
- Λογισμικό Οδήγησης του Ελεγκτή του Host *(Host Controller Driver, HCD)*: Το interface λογισμικού μεταξύ του Host Controller του USB και του USB System Software.

Η υλοποίηση του USBD εξαρτάται από το λειτουργικό σύστημα του Host, και παρέχει επικοινωνία μεταξύ του λογισμικού του Host *(Host Software)* και των Clients, παρέχοντας σε αυτά τις απαραίτητες μεθόδους για τη διαχείριση των USB συσκευών. Το HCD εξαρτάται από την συγκεκριμένη υλοποιήση του HC που οδηγεί, και επιτρέπει την ανεξαρτησία των υπόλοιπων στρωμάτων του Host από τις διάφορες

δυνατές υλοποιήσεις του HC. Στο Σχήμα 3.1.1 παρουσιάζεται λεπτομερώς η δομή του Host ενός συστήματος USB:



Σχήμα 3.1.1: Λεπτομερής Παρουσιάση της Δομής του USB Host

Στις επόμενες παραγράφους αναλύονται εκτενέστερα τα βασικά μέρη του USB Host, χωρίς να δίνονται λεπτομέρειες για τον τρόπο υλοποίησης τους. Ο αναγνώστης μπορεί να βρει περισσότερες λεπτομέρειες στο Universal Serial Bus Specification Revision 1.1 **[1]**. Περιγράφεται ο ρόλος του κάθε μέρους του USB και ο τρόπος με τον οποίο επικοινωνεί με τα γειτονικά του μέρη του USB. Όπως φαίνεται στο παραπάνω σχήμα, το στρώμα Λογισμικού του Συστήματος USB *(USB System Software)* αποτελείται από το HCD, το USBD, και το Λογισμικό του Host *(Host Software)*. Το τελευταίο αποτελεί το λειτουργικό σύστημα του Host, εάν υπάρχει, και δεν αποτελεί βασικό μέρος του USB, απλώς παρέχει υποστήριξη στο USB. Για τον λόγο αυτό δεν αναλύεται παρακάτω.

3.1.1 Λογισμικό Συσκευής (Client Software)

Το Client SW καθορίζει τις μεταφορές δεδομένων που πρέπει να γίνουν για ένα function, μέσω των IRP. Το Client SW γνωρίζει μόνο τα pipes που αντιστοιχούν στο συγκεκριμένο function που διαχειρίζεται. Ένα Client SW δεν παραβιάζει τους περιορισμούς πρόσβασης στον δίαυλο και χρησιμοποίησης του εύρους ζώνης που αντιστοιχούν σε κάθε pipe που χρησιμοποιεί. Επικοινωνεί με το USB System Software μέσω του USBD interface, στέλνοντας τα IRP του function που διαχειρίζεται.

Μερικά Clients μπορεί να διαχειρίζονται τα functions τους χρησιμοποιώντας διαπροσωπείες ορισμένες από το λειτουργικό συστήμα και όχι το USBD. Όμως, και σε αυτή την περίπτωση το λειτουργικό σύστημα θα χρησιμοποιεί κάποιο Client κατώτερου επιπέδου για να περάσει τα απαραίτητα IRP στο USBD.

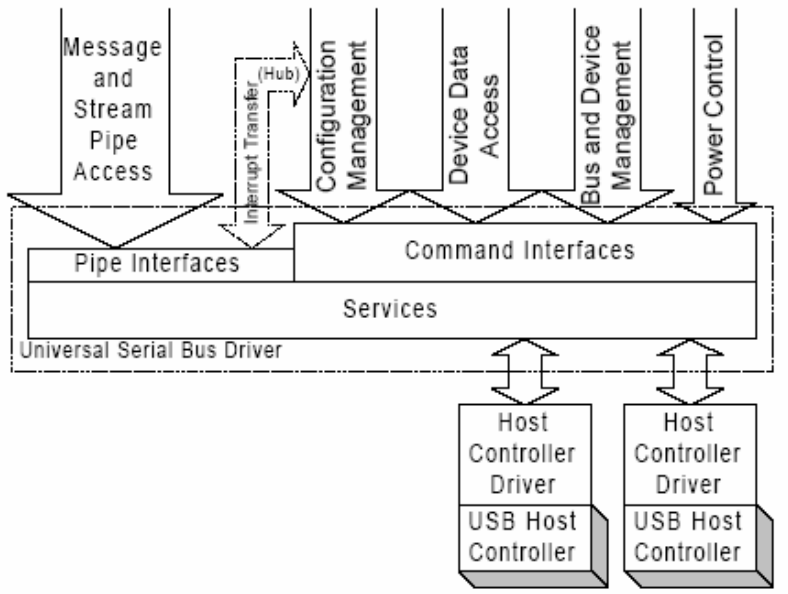
3.1.2 Λογισμικό Οδήγησης του USB (USB Driver)

Το USBD παρεμβαίνει στη λειτουργία του USB Host για τη ρύθμιση των συσκευών που συνδέονται στο USB, και για την πραγματοποίηση των μεταφορών δεδομένων που αιτούνται από τα Clients.

Όταν μια συσκευή συνδέεται και ρυθμίζεται, το USBD λαμβάνει αιτήσεις ρύθμισης από το κατάλληλο λογισμικό, που περιγράφουν την επιθυμητή ρύθμιση της συσκευής: Τον αριθμό του endpoint, τον τύπο μεταφοράς δεδομένων που υποστηρίζει το endpoint, το μέγιστο data payload του endpoint, κτλ. Το USBD δέχεται ή απορρίπτει μία αίτηση ρύθμισης με βάση τη διαθεσιμότητα του εύρος ζώνης του διαύλου και την ικανότητα υποστήριξης του συγκεκριμένου τύπου μεταφοράς δεδομένων. Αν η αίτηση γίνει δεκτή, το USBD δημιουργεί το κατάλληλο pipe με βάση τις πληροφορίες που έχει λάβει από την αίτηση. Μετά την επιτυχή ρύθμιση της συσκευής, το Client μπορεί να στέλνει IRP στο USBD. Το USBD παραδίδει τα IRP στα κατώτερα στρώματα του Host, και επιστρέφει στα Clients το status της εκτέλεσής τους.

Η υλοποίηση του USBD εξαρτάται από το λειτουργικό σύστημα του Host. Επομένως, τα Clients πρέπει να υλοποιούνται έτσι ώστε να είναι συμβατά με τη διαπροσωπεία που παρέχει προς αυτά το USBD *(USB Driver Interface, USBDI)*. Το USBD παρέχει στα Clients μηχανισμούς λογισμικού, που διακρίνονται σε δύο ομάδες: Μηχανισμούς Εντολών *(Command Mechanisms)*, και Μηχανισμούς εξυπηρέτησης των Pipes *(Pipe Mechanisms)*. Οι μηχανισμοί εντολών επιτρέπουν στα Clients να ρυθμίζουν και να ελέγχουν την λειτουργία του USBD και της USB συσκευής που οδηγούν, χρησιμοποιώντας το Default Control Pipe κάθε συσκευής. Οι μηχανισμοί εξυπηρέτησης των pipes επιτρέπουν στα Clients την διαχείριση των υπόλοιπων pipes κάθε συσκευής. Γενικά, το USBD είναι υπεύθυνο για τον συνολικό έλεγχο του συστήματος USB. Έτσι, έχει πρόσθετες αρμοδιότητες, όπως η αντιμετώπιση σφαλμάτων, η διαχείριση της αποσύνδεσης των συσκευών, η διαχείριση των IRP (τοποθέτηση σε ουρές, απόσυρση, επιστροφή κατάστασης), η η διαχείριση της τροφοδότησης ισχύος των μη αυτοτροφοδοτούμενων συσκευών, η

ανάθεση διευθύνσεων στις συσκευές, κ.α. Στο παρακάτω σχήμα φαίνεται η συνολική εικόνα του USBD.



Σχήμα 3.1.2: Συνολική Εικόνα του USBD

3.1.3 Λογισμικό Οδήγησης του Ελεγκτή του Host (Host Controller Driver)

Το HCD αποτελεί το κατώτερο στρώμα της στοίβας λογισμικού του Host. Μόνο το HCD επικοινωνεί άμεσα με τον HC που οδηγεί. Η υλοποίησή του εξαρτάται από την υλοποίηση του HC που οδηγεί (UHCI ή OpenHCI). Το HCD μπορεί να διαχειρίζεται περισσότερους από έναν HC, αποκρύπτοντας τις λεπτομέρειες υλοποίησής τους από τα ανώτερα στρώματα του USB Host. Το HCD επικοινωνεί «προς τα πάνω» με το USBD. Σε κάθε λειτουργικό σύστημα το interface που παρέχει το HCD προς το USBD είναι τυποποιημένο, και αναφέρεται ως HCDI *(Host Controller Driver Interface)*.

Το HCD λαμβάνει από το USBD τα IRP προς εκτέλεση και τα παραδίδει στο HC για εκτέλεση. Αυτό γίνεται μέσω μιας δομής δεδομένων που αναφέρεται ως Transaction List *(Λίστα Συναλλαγών)*, η οποία είναι προσπελάσιμη από το υλικό και το λογισμικό. Η μορφή του Transaction List εξαρτάται από την υλοποίηση του interface υλικού-λογισμικού (UHCI ή OpenHCI). Το HCD, συμπληρώνει το Transaction List με βάση τις πληροφορίες που περιέχουν τα IRP, σε μορφή αναγνώσιμη από το HC.

Ανάλογα με την υλοποίηση του HC, το HCD μπορεί να χρειάζεται να «τεμαχίζει» τα IRP σε μικρότερες αιτήσεις οι οποίες ονομάζονται URB *(USB Request Block)*, έτσι ώστε κάθε URB να αντιστοιχεί σε ένα μόνο transaction, πριν από την τοποθέτηση στο Transaction List. Σε αντίθετη περίπτωση η εργασία αυτή γίνεται αυτόματα από το υλικό μετά την ανάγνωση του Transation List. Επίσης, ανάλογα με την υλοποίηση, η χρονοδρομολλόγηση των αιτήσεων μπορεί να γίνεται αυτόματα από

το υλικό ή να είναι ευθύνη του HCD. Τα παραπάνω θέματα αναλύονται με μεγαλύτερη λεπτομέρεια στην ενότητα 3.2.

Το HC εκτελεί τα transactions που υποδεικνύει το Transaction List και ενημερώνει τα πεδία του τελευταία με το status των transactions. Με βάση το status το HCD ενημερώνει τα IRP για τον κατάλληλο χειρισμό τους από το USBD.

Σε κάθε υλοποίηση, το HCD είναι επίσης υπεύθυνο για τον χειρισμό του HC, με βάση τις υποδείξεις που δέχεται από το USBD, αφού είναι η μοναδική οντότητα λογισμικού που επικοινωνεί άμεσα με τον HC. Για τον λόγο αυτό το HCD παρέχει μια αφαιρετική εικόνα του HC στο USBD, μέσω διαφόρων δομών δεδομένων. Το HCD διαχειρίζεται το υλικό μέσω των καταχωρητών που διαθέτει ο HC.

3.1.4 Ελεγκτής του Host (Host Controller)

Ο HC αποτελεί το κατώτερο στρώμα του USB Host και αποτελεί το υλικό που συνδέεται άμεσα στο USB, πραγματοποιώντας όλες τις μεταφορές δεδομένων που αιτούνται από το λογισμικό. Για την διαχείρισή του και την επικοινωνία με το λογισμικό, ο HC παρέχει προς το HCD μια διαπροσωπεία υλικού. Υπάρχουν δύο τυποποιημένες διαπροσωπείες του HC, ανάλογα με την υλοποίησή του: Το Universal Host Controller Interface *(UHCI)* και το Open Host Controller Interface *(OpenHCI)*. Οι διαπροσωπείες αυτές υλοποιούνται με διάφορους καταχωρητές του HC. Πληροφορίες για τις διαπροσωπείες αυτές δίνονται στην επόμενη ενότητα.

Ο HC διαβάζει τις πληροφορίες που περιέχει το Transaction List, και μεταφράζει τις πληροφορίες αυτές σε transactions στον δίαυλο ακολουθώντας το πρωτόκολλο USB. Μετά την ολοκλήρωση ενός transaction, επιτυχώς ή ανεπιτυχώς, ο HC αναφέρει το αντίστοιχο status στο HCD, ενημερώνοντας κατάλληλα το Transaction List και τους καταχωρητές του.

Κάθε HC πρέπει να υποστηρίζει τις παρακάτω βασικές λειτουργίες (τουλάχιστον):

- Να αναφέρει την κατάστασή του στο HCD και να παρέχει μεθόδους χειρισμού του από το λογισμικό.
- Να στέλνει στον δίαυλο σε σειριακή μορφή τα δεδομένα προς αποστολή, τα οποία βρίσκονται σε buffers της μνήμης που υποδεικνύει το HCD, και να πραγματοποιεί την αντίστροφη διαδικασία όταν λαμβάνει δεδομένα από το USB.
- Να παράγει και να στέλνει πακέτα SOF στον δίαυλο αυτόματα κάθε 1 ms.
- Να επεξεργάζεται τα δεδομένα που λαμβάνει από το HCD, μέσω του Transaction List, και με βάση αυτά να εκτελεί τα transactions.
- Να υποστηρίζει το πρωτόκολλο διαύλου USB.
- Να αντιμετωπίζει τα σφάλματα που ανιχνεύει όπως ορίζεται από το USB.

- Να μπορεί να θέσει το USB σε ανενεργή κατάσταση (Suspend State), και να το ενεργοποιεί όταν λάβει την κατάλληλη εντολή.

- Να υλοποιεί τον κεντρικό διακλαδωτή *(Root Hub)* του συστήματος USB.

- Να παρέχει μία πλήρη διαπροσωπεία προς το HCD, έτσι ώστε να είναι δυνατός ο χειρισμός του από το λογισμικό του Host.

3.2 Διαπροσωπείες Ελεγκτή του Host

Σε αυτή την ενότητα δίνονται γενικές πληροφορίες για τις δύο τυποποιημένες διαπροσωπείες του HC προς το HCD, που είναι συμβατές με το πρωτόκολλο USB 1.1. Οι διαπροσωπείες αυτές είναι οι Universal Host Controller Interface *(UHCI)* και Open Host Controller Interface *(OpenHCI)*. Η περιγραφή που ακολουθεί δεν δίνει τεχνικές λεπτομέρειες σχετικά με την υλοποίηση των διαπροσωπειών αυτών, και των αντίστοιχων HC. Καταγράφονται μόνο τα γενικά χαρακτηριστικά τους. Το UHCI περιγράφεται λεπτομερώς στο Κεφάλαιο 4 και τα παραρτήματα Α και Β του παρόντος συγγράμματος. Επίσης, για μια πλήρη περιγραφή των διαπροσωπειών, ο αναγνώστης καλείται να ανατρέξει στις προδιαγραφές του UHCI **[2]** και OpenHCI **[3]**. Τέλος, για λόγους πληρότητας σημειώνεται ότι το πρωτόκολλο USB 2.0 υποστηρίζεται μέσω της διαπροσωπείας Enhanced Host Controller Interface *(EHCI)* της Intel **[4]**.

3.2.1 Σύγκριση των UHCI και OpenHCI

Το UHCI αποτελεί το παλαιότερο και δημοφιλέστερο Interface υλικού-λογισμικού συμβατό με το USB 1.1. Οι προδιαγραφές του τέθηκαν από την εταιρία Intel το 1996, ταυτόχρονα με την ανάπτυξη του πρωτοκόλλου USB, με κύριο στόχο την όσο το δυνατόν μικρότερη πολυπλοκότητα του υλικού (HC), επιβαρύνοντας όμως το λογισμικό (HCD) και την κεντρική μόναδα επεξεργασίας *(Central Processing Unit, CPU)* του Host με πολύπλοκες επεξεργασίες για την οδήγησή του και την χρονοδρομολλόγησή των αιτήσεων για τη μεταφορά δεδομένων.

Οι προδιαγραφές του OpenHCI τέθηκαν το 1999 από τις εταιρίες Compaq, Microsoft και National Semiconductor. Σε σύγκριση με το UHCI, το OpenHCI επιτυγχάνει την μείωση της πολυπλοκότητας του HCD, και την αποφόρτιση της CPU του Host από πολύπλοκες επεξεργασίες, αναθέτοντας στο υλικό (HC) την πραγματοποίηση κάποιων επεξεργασιών που στο UHCI γίνονται από το HCD, αυξάνοντας όμως κατά πολύ την πολυπλοκότητα του υλικού. Επιπλέον, το OpenHCI υποστηρίζει άμεση προσπέλαση της κύριας μνήμης, μέσω μιας μονάδας DMA *(Direct Memory Access)*, αποδεσμεύοντας την CPU από την προσπέλαση της μνήμης. Το OpenHCI είναι πιο ισορροπημένο σε σχέση με το UHCI σε ό,τι αφορά την πολυπλοκότητα του υλικού και του λογισμικού.

Σε κάθε περίπτωση το USBD του Host θα είναι ανεξάρτητο της υλοποίησης του HC που χρησιμοποιείται, αφού για κάθε λειτουργικό σύστημα η μορφή του HCDI είναι καθορισμένη. Άλλωστε, το HCD αποκρύπτει από τα ανώτερα στρώματα του USB Host τις λεπτομέρειες της υλοποίησης του HC. Επίσης οι USB συσκευές είναι ανεξάρτητες από την υλοποίηση του HC από τον οποίο διαχειρίζονται.

Από όσα έχουν ήδη αναφερθεί φαίνεται ότι κάθε μία υλοποίηση έχει τα πλεονεκτήματα και τα μειονεκτήματά της έναντι της άλλης. Βέβαια, και οι δύο υλοποιήσεις ικανοποιούν τις προδιαγραφές του πρωτοκόλλου USB και έχουν τις ίδιες δυνατότητες. Οι ουσιώδεις διαφορές τους εντοπίζονται στην κατανομή του «φόρτου εργασίας» στο υλικό και το λογισμικό, στο κόστος υλοποίησης του υλικού, στην

επιβάρυνση της CPU του Host, και στην ταχύτητα πρόσβασης των καταχωρητών του υλικού. Παρακάτω επιχειρείται μια σύγκριση των κύριων χαρακτηριστικών των δύο διαπροσωπειών.

3.2.1.1 Πολυπλοκότητα Υλικού και Κόστος Υλοποίησης

Το UHCI απαιτεί από το υλικό να πραγματοποιεί μόνο τις βασικές απαιτούμενες λειτουργίες ενός HC που περιγράφτηκαν στην ενότητα 3.1.4. Έτσι η πολυπλοκότητα του υλικού είναι περιορισμένη, της τάξης των δέκα χιλιάδων πυλών, με αποτέλεσμα την ευκολία υλοποίησης του HC με εξαιρετικά χάμηλο κόστος για τον κατασκευαστή και τον καταναλωτή. Το ολοκληρωμένο κύκλωμα *(Chip)* που υλοποιεί τον ελεγκτή έχει μικρό αριθμό ακροδεκτών, με αποτέλεσμα την εύκολη προσαρμογή του σε ευρύτερα ολοκληρωμένα κυκλώματα *(Chipsets)*.

Αντίθετα, το OpenHCI απαιτεί από το υλικό την υλοποίηση πρόσθετων πολύπλοκων λειτουργιών, με αποτέλεσμα την αυξημένη πολυπλοκότητά του . Ένας ελεγκτής συμβατός με το OpenHCI πρέπει να υλοποιεί έναν αλγόριθμο χρονοδρομολόγησης των αιτήσεων μεταφοράς δεδομένων, συμβατό με τις απαιτήσεις του USB (βλέπε Ενότητες 2.2.5 έως 2.2.8). Επίσης, το υλικό έχει την ευθύνη της αποκωδικοποίησης των πληροφοριών των IRP σε transactions. Σε έναν ελεγκτή UHCI οι παραπάνω λειτουργίες πραγματοποιούνται από το HCD. Η αυξημένη πολυπλοκότητα των ελεγκτών OpenHCI, της τάξης των πενήντα έως εβδομήντα χιλιάδων πυλών, έχει ως συνέπεια το αυξημένο κόστος υλοποίησής τους. Παρ'όλα αυτά, τα Chips που υλοποιούν τους HC έχουν σχετικά μικρό αριθμό ακροδεκτών, με αποτέλεσμα την εύκολη προσαρμογή τους σε Chipsets.

3.2.1.2 Ευελιξία της Υλοποίησης

Λόγω των περιορισμένων αρμοδιοτήτων του υλικού, το UHCI δίνει στον προγραμματιστή περισσότερη ελευθερία στην υλοποίηση του HCD, προσαρμόζοντας τη λειτουργία του USB Host στις ανάγκες μιας συγκεκριμένης εφαρμογής, σε σχέση με το OpenHCI στο οποίο πολλές πρόσθετες λειτουργίες εκτελούνται αυτόματα από το υλικό και δεν υπάρχει δυνατότητα παρέμβασης στον τρόπο πραγματοποίησής τους.

3.2.1.3 Υλοποίηση των Καταχωρητών της Διαπροσωπείας

Κάθε διαπροσωπεία υλικού-λογισμικού διαθέτει ένα σύνολο καταχωρητών για την επικοινωνία του HC με το HCD και την διαχείριση του HC από το λογισμικό του Host. Ωστώσο, ο τρόπος υλοποίησης των καταχωρητών στο UHCI και το OpenHCI διαφέρει. Οι καταχωρητές του OpenHCI αντιστοιχούν σε διευθύνσεις της κύριας μνήμης *(Memory Mapped)* του Host, ενώ οι καταχωρητές του UHCI σε διευθύνσεις Εισόδου/Εξόδου *(I/O Space)* του Host. Αυτό αποτελεί μειονέκτημα για το UHCI, γιατί η κύρια μνήμη προσπελάσσεται ταχύτερα σε σχέση με το I/O Space ενός υπολογιστικού συστήματος. Επίσης, το μέγεθος του I/O Space είναι περιορισμένο σε

σχέση με το μέγεθος της κύριας μνήμης. Σε μελλοντικές υλοποιήσεις ελεγκτών USB, οι οποίες θα είναι ακόμα πιο πολύπλοκες, το I/O Space θα είναι ανεπαρκές για την τοποθέτηση των καταχωρητών τους.

3.2.1.4 Πολυπλοκότητα Λογισμικού και Απασχόληση της CPU του Host

Στο OpenHCI η χρονοδρομολόγηση γίνεται τοποθετώντας τις αιτήσεις σε ουρές. Το Transaction List του OpenHCI διαθέτει ξεχωριστές ουρές για τις περιοδικές μεταφορές δεδομένων (Isochronous και Interrupt), τις μεταφορές ελέγχου (Control), και τις ογκώδεις μεταφορές (Bulk), παρέχοντας με αυτόν τον τρόπο ένα απλό interface για το λογισμικό. Το υλικό διαθέτει έναν εξεζητημένο αλγόριθμο αυτόματης χρονοδρομολόγησης και εκτέλεσης των αιτήσεων που λαμβάνονται από το Transaction List. Το υλικό του OpenHCI αναλαμβάνει τον «τεμαχισμό» των πληροφοριών που βρίσκει στο Transaction List σε στοιχειώδη transactions, και την εκτέλεση του Schedule των transactions στον δίαυλο.

Αντίθετα, στο UHCI οι παραπάνω λειτουργίες εκτελούνται από το HCD. Το HCD αναλαμβάνει τον τεμαχισμό των IRP σε URB, αιτήσεις που κάθε μια αντιστοιχεί σε ένα μόνο transaction, και την κατάλληλη τοποθέτησή τους στο Transaction List της διαπροσωπείας για την επιθυμητή χρονοδρομολόγηση των transactions στον δίαυλο. Ενώ στο OpenHCI οι αιτήσεις τοποθετούνται με σειρά FIFO στις ουρές του Transaction List, το Transaction List UHCI δεν διαθέτει έτοιμες ουρές και αυτές πρέπει να υλοποιηθουν από το λογισμικό. Η τοποθέτηση των transactions στις ουρές του UHCI δεν είναι τόσο απλή όσο στο OpenHCI, γιατί η θέση κάθε transaction στο Transaction List ορίζει και τη σειρά χρονοδρομολόγησης του στον δίαυλο. Το HCD πρέπει να υπολογίζει τον αναμενόμενο χρόνο που θα διαρκέσει ένα transaction στον δίαυλο για την τήρηση του περιορισμένου χρόνου κάθε frame και την κατάλληλη εξυπηρέτηση των διαφορετικών τύπων μεταφοράς δεδομένων. Στην Ενότητα 3.2.2 δίνονται πληροφορίες για την μέθοδο τεμαχισμού των IRP σε URB και τον υπολογισμό της αναμενόμενης διάρκειας ενός transaction. Στο Κεφάλαιο 4 αναλύεται πλήρως το πρότυπο της διαπροσωπείας UHCI.

Σαν συνέπεια των παραπάνω, το UHCI απαιτεί ένα πολύπλοκο HCD για την οδήγησή του HC, το οποίο απασχολεί σε σχετικά μεγάλο βαθμό την CPU του συστήματος. Αντίθετα, το OpenHCI οδηγείται από ένα σχετικά ένα απλό και εύκολα υλοποιήσιμο HCD. Το υλικό αναλαμβάνει το ίδιο την πραγματοποίηση πολύπλοκων επεξεργασιών χωρίς να απασχολεί πολύ την CPU. Επίσης το OpenHCI υποστηρίζει μια μονάδα DMA για την προσπέλαση της κεντρικής μνήμης, αποφορτίζοντας περεταίρω την CPU, κάτι το οποίο δεν συμβαίνει στο UHCI.

3.2.1.5 Συμπέρασμα

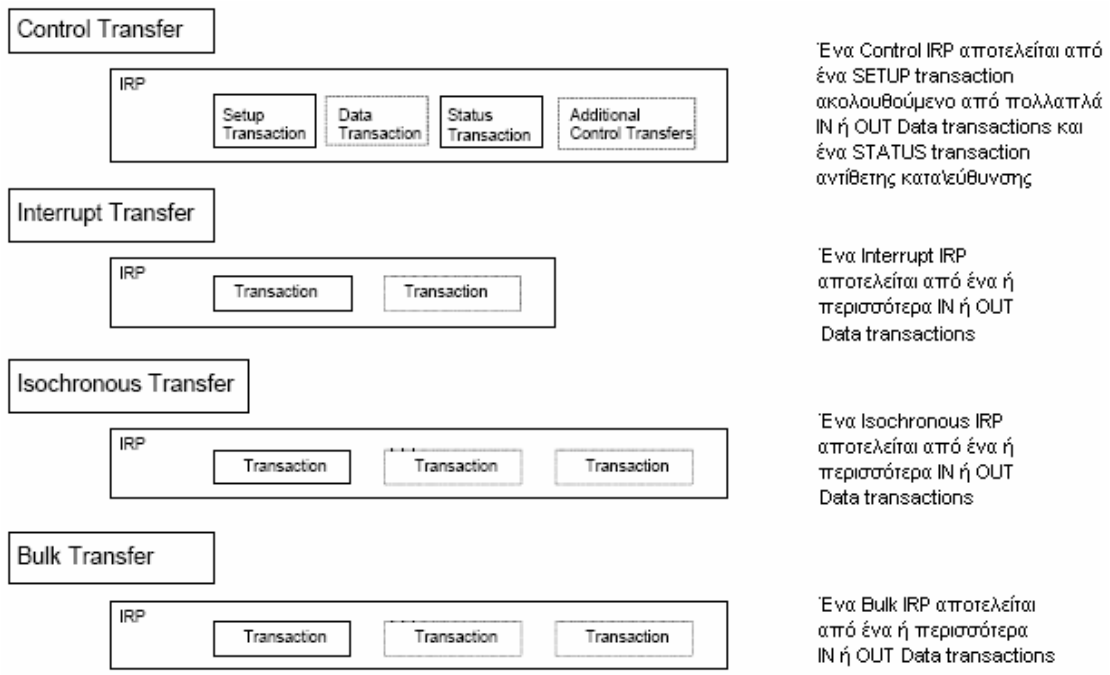
Από τα παραπάνω εύκολα συμπεραίνεται ότι το OpenHCI διαθέτει πλήθος πλεονεκτημάτων έναντι του UHCI. Παρ'όλα αυτά, η υλοποίηση αξιόπιστων οδηγών για UHCI Host Controllers που βελτιστοποιούν την απόδοση του συστήματος, σε συνδυασμό με το χαμηλό κόστος υλοποίησης και διάθεσης σε σχέση με τους ελεγκτές OpenHCI, έχουν καταστήσει το UHCI ως το πιο δημοφιλές πρότυπο διαπροσωπείας υλικού-λογισμικού του USB Host για το πρωτόκολλο USB 1.1. Ένας ακόμα λόγος για την εμπορική επικράτηση του UHCI είναι ότι ελεγκτές UHCI διατίθονται σε Chipsets ελεγκτών EHCI, που υποστηρίζουν το πρωτόκολλο USB 2.0 και είναι απαραίτητοι σε κάθε υπολογιστικό σύστημα που απαιτεί υψηλές ταχύτητες μεταφοράς δεδομένων. Αντίθετα, οι εταιρίες που κατασκευάζουν ελεγκτές OpenHCI δεν έχουν κατασκευάσει ακόμα ελεγκτές που να είναι συμβατοί με το πρωτόκολλο USB 2.0. Η ευρύτητα και η διάδοση του UHCI αποτελεί τον λόγο για τον οποίο επιλέχθηκε η υλοποίηση του λογισμικού οδήγησής του στα πλαίσια της παρούσας διπλωματικής εργασίας.

3.2.2 Χρονοδρομολόγηση των Αιτήσεων Μεταφοράς Δεδομένων

Όπως έχει ήδη αναφερθεί, στο UHCI ο λογισμικό είναι υπεύθυνο για τον τεμαχισμό των αιτήσεων IRP, που αντιστοιχούν σε πολλά transactions, σε URB που αντιστοιχούν σε ένα μόνο transaction του IRP. Επίσης, το λογισμικό είναι υπεύθυνο για την χρονοδρομολόγηση των transactions, εξυπηρετώντας όλους τους τύπους μεταφοράς δεδομένων όπως ορίζεται από το USB μέσα στο διαθέσιμο χρόνο ενός frame. Για τον λόγο αυτό το HCD υπολογίζει το μέγιστο αναμενόμενο χρόνο κάθε transaction (εάν αυτό πραγματοποιηθεί χωρίς σφάλματα). Στο OpenHCI οι παραπάνω επεξεργασίες γίνονται από το υλικό και δεν απασχολούν τον προγραμματιστή. Παρακάτω περιγράφεται ο τρόπος με τον οποίο το HCD του UHCI πραγματοποιεί τις παρακάτω λειτουργίες. Παρόμοια, αλλά σε επίπεδο υλικού, πραγματοποιούνται οι λειτουργίες αυτές στο OpenHCI. Η λεπτομερής περιγραφή της χρονοδρομολόγησης *(Scheduling)* στο UHCI γίνεται στο Κεφάλαιο 4.

Στην περίπτωση που ο HC που οδηγεί το HCD είναι συμβατός με το πρότυπο UHCI, το HCD που το οδηγεί λαμβάνει τα IRP από το USBD και τα «τεμαχίζει» σε μικρότερες αιτήσεις οι οποίες αναφέρονται ως URB *(USB Request Block)*, όπως φαίνεται στο Σχήμα 3.2.1, με βάση τις πληροφορίες που περιέχουν τα IRP. Έτσι, το HCD αναλύει κάθε αίτηση στα transactions που την συνθέτουν, ανάλογα με τον τύπο μεταφοράς του pipe στο οποίο ανήκει το IRP, όπως αναλύεται στην Ενότητα 2.3.4.

Ο υπολογισμός της μέγιστης αναμενόμενης διάρκειας ενός transaction γίνεται χρησιμοποιώντας τις παρακάτω εξισώσεις, ανάλογα με την ταχύτητα και τον τύπο μεταφοράς του transaction, οι οποίες ορίζονται στο USB Specification 1.1:



Σχήμα 3.2.1: Τεμαχισμός των IRP σε URB

Full-speed IN Transactions

Non-Isochronous Transfer (Περιλαμβανομένου του Handshake):
Bus_Time= 9107+ (83.54· Floor(3.167+ BitStuffTime(Data_bc)))+ Host_Delay *(3.2.1)*

Isochronous Transfer (Χωρίς Handshake):
Bus_Time= 7268+ (83.54· Floor(3.167+ BitStuffTime(Data_bc)))+ Host_Delay *(3.2.2)*

Full-speed OUT Transactions

Non-Isochronous Transfer (Περιλαμβανομένου του Handshake):
Bus_Time= 9107+ (83.54· Floor(3.167+ BitStuffTime(Data_bc)))+ Host_Delay *(3.2.3)*

Isochronous Transfer (Χωρίς Handshake):
Bus_Time= 6265+ (83.54· Floor(3.167+ BitStuffTime(Data_bc)))+ Host_Delay *(3.2.4)*

Low-speed IN Transactions

Bus_Time= 64060+ (2· Hub_LS_Setup)+ (676.67· Floor(3.167+ BitStuffTime(Data_bc))) + Host_Delay *(3.2.5)*

Low-speed OUT Transactions

Bus_Time= 64107+ (2· Hub_LS_Setup)+ (667.0· Floor(3.167+ BitStuffTime(Data_bc))) + Host_Delay *(3.2.6)*

Το αποτέλεσμα των παραπάνω τύπων εκφράζεται σε nanoseconds. Παρακάτω εξηγείται ο συμβολισμός που χρησιμοποιείται:

- *Data_bc*: Η τιμή του Data Payload του πακέτου δεδομένων του transaction.
- *Host_Delay*: Ο χρονική καθυστέρηση του HC για την πραγματοποίηση του transaction σε nanoseconds. Εξαρτάται από την υλοποίηση του HC.
- *Floor()*: Συνάρτηση που επιστρέφει το ακέραιο μέρος του ορίσματός της.
- *Hub_LS_Setup*: Το χρονικό διάστημα κατά το οποίο ο HC παραμένει ανενεργός, έτσι ώστε τα Hubs να ενεργοποιήσουν τις low-speed θύρες, σε nanoseconds. Στην καλύτερη περίπτωση ισούται με χρόνο τεσσάρων full-speed bits.
- *BitStuffTime()*: Συνάρτηση που υπολογίζει την χρονική καθυστέρηση που προστίθεται λόγω της εφαρμογής του Bit Stuffing στα δεδομένα. Στην χειρότερη περίπτωση ισοδυναμεί με 1.1667· 8· Data_bc.

Σε κάθε υλοποίηση το υλικό ενημερώνει το Transaction List με το αποτέλεσμα *(status)* της εκτέλεσης κάθε transaction. Το HCD λαμβάνει τις πληροφορίες αυτές από το Transaction List και τις παραδίδει στο USBD για τον κατάλληλο χειρισμό των IRP.

3.2.3 Διαθέσιμες Υλοποιήσεις

Στο εμπόριο διατίθονται πολλές υλοποιήσεις ελεγκτών συμβατών με τα πρότυπα UHCI ή OpenHCI, αλλά και το πρότυπο EHCI. Το πρότυπο UHCI υποστηρίζεται από ολοκληρωμένα κυκλώματα (Chips ή Chipsets) κυρίως των εταιριών Intel και VIA. Chips συμβατά με το πρότυπο OpenHCI κατασκευάζονται από τις εταιρίες SiS, ALi, Opti, Agere Systems, και Silicon Core. Το πρότυπο EHCI υποστηρίζεται από ελεγκτές των εταιριών Intel, VIA και NEC. Στον παρακάτω πίνακα παρουσιάζονται μερικές από τις πιο δημοφιλείς υλοποιήσεις ελεγκτών USB Host για κάθε πρότυπο. Βέβαια, ο αριθμός των διαθέσιμων υλοποιήσεων είναι πολύ μεγαλύτερος.

Πίνακας 3.2.1: Διαθέσιμες Υλοποιήσεις Ελεγκτών USB Host

USB 1.1 UHCI	USB 1.1 OpenHCI	USB 2.0 EHCI
Intel 440BX	CMD USB0670	Intel 82865PE (MCH)
Intel 440LX	CMD USB0673	Intel i82801DB ICH4
Intel i82371AB southbridge (PIIX4)	Lucent USS-302	Intel i82801DBM ICH4-M
Intel i82371EB southbridge (PIIX4E)	nVidia nForce	Intel Springdale-G 865G ICH5
Intel i82801AA ICH	OPTi 82C861	NEC μ7201 family
Intel i82801CA ICH3-S	SiS SiS630S southbridge	VIA VT6202
Intel i82801CAM ICH3-M	SiS7001	
VIA VT83C572	SiS735 southbridge	

VIA VT82C596B southbridge	SiS745 southbridge	
VIA VT82C686A southbridge		
VIA VT82C686B southbridge		

ΚΕΦΑΛΑΙΟ 4

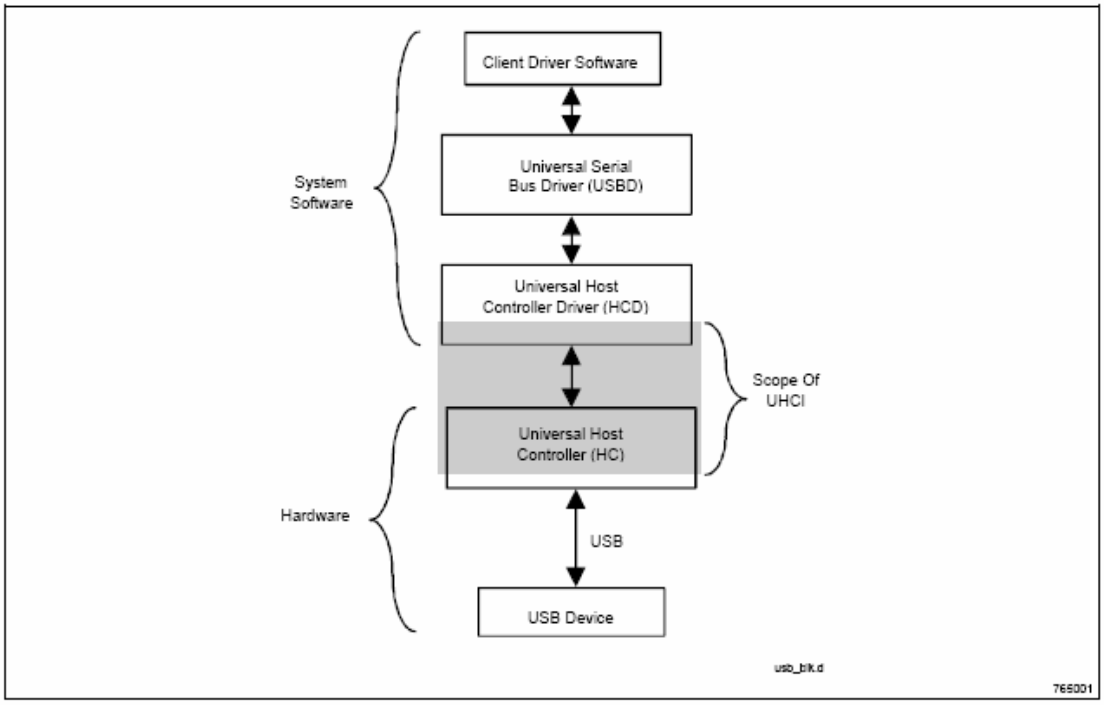
Universal Host Controller Interface (UHCI)

Στο κεφάλαιο αυτό περιγράφεται το Universal Host Controller Interface *(UHCI)* για μια συσκευή που υλοποιεί τον Universal Serial Bus Host Controller της Intel. Πρόκειται για τη διαπροσωπεία *(interface)* που συνδέει το Host Controller Software Driver *(HCD)* και το hardware του Host Controller *(HC)*. Ο Host Controller υλοποιεί το πρωτόκολλο USB 1.1, το οποίο περιγράφηκε στο προηγούμενο κεφάλαιο. Η ανάλυση που ακολουθεί βασίζεται στις προδιαγραφές της Intel για το Universal Host Controller Interface **[2]** και αφορά κυρίως την πλευρά του υλικού του interface και τους περιορισμούς του λογισμικού, ώστε να ακολουθεί τις προδιαγραφές του UHCI. Το λογισμικό που γράφτηκε για το UHCI, δηλαδή το HCD, παρουσιάζεται σε επόμενο κεφάλαιο. Αρχικά παρουσιάζονται η αρχιτεκτονική και τα βασικά στοιχεία του τρόπου λειτουργίας του UHCI και στη συνέχεια ακολουθεί λεπτομερής ανάλυση των καταχωρητών και των διακοπών υλικού του Host Controller, των δομών δεδομένων που χρησιμοποιούνται για την περιγραφή των αιτήσεων μετάδοσης δεδομένων στο USB, και του τρόπου με τον οποίο ο Host Controller χειρίζεται τις αιτήσεις αυτές *(Χρονοδρομολόγηση, Scheduling)*.

4.1 Συνολική Εικόνα του Universal Host Controller Interface

4.1.1 Υλικό και Λογισμικό του UHCI

Στο σχήμα 4.1 παρουσιάζεται η δομή ενός συστήματος USB Host, όπως έχει περιγραφτεί στο Κεφάλαιο 2. Το σύστημα αυτό χρησιμοποιεί ένα Host Controller συμβατό με το πρότυπο Universal Host Controller Interface. Όπως φαίνεται στο σχήμα, η διαπροσωπεία UHCI παρέχει επικοινωνία μεταξύ του HC και του λογισμικού οδήγησής του, δηλαδη του HCD.



Σχήμα 4.1.1: Block διάγραμμα του συστήματος USB Host

Το HCD εξασφαλίζει την επικοινωνία μεταξύ του HC και του USBD και κατ'επέκταση του λειτουργικού συστήματος, μεταφράζοντας τις αιτήσεις από το USBD για μεταφορά δεδομένων και δημιουργώντας τις κατάλληλες δομές δεδομένων για την επικοινωνία με τον HC. Οι δομές αυτές καταλαμβάνουν χώρο στην κύρια μνήμη και περιέχουν όλες τις απαραίτητες πληροφορίες για την επικοινωνία μεταξύ του Client Software στον Host, και της συσκευής που συνδέεται σε αυτόν μέσω του USB. Μετά την περάτωση των συναλλαγών (*transaction*s) στο USB, το HCD πρέπει να αναφέρει στα ανώτερα στρώματα του συστήματος USB το αποτέλεσμα *(status)* της μεταφοράς δεδομένων, για να το χειριστούν αυτά κατάλληλα. Οι δομές που χρησιμοποιεί το HCD αλλά και ο HC είναι οι Frame List, Transfer Descriptor, Queue Head και Data Buffer. Τέλος το HCD μπορεί να γράφει και να διαβάζει τους καταχωρητές του HC ώστε να καθορίζει τις διάφορες παραμέτρους του HC και να ελέγχει το status του.

Ο HC αποτελεί το μέρος του υλικού του interface και επεξεργάζεται τις παραπάνω δομές δεδομένων που βρίσκονται στην κύρια μνήμη του Host για να εκτελέσει το κατάλληλο transaction στο USB. Το HCD οδηγεί τον HC παρέχοντάς του τις απαραίτητες πληροφορίες για τις λειτουργίες που πρέπει να εκτελέσει και τις αιτήσεις για transactions, και ο HC τις εκτελεί ακολουθώντας το πρωτόκολλο USB και αναφέροντας το status μετά την εκτέλεσή τους. Επίσης το HCD εξυπηρετεί τις διακοπές που εγείρει ο HC. Σε περίπτωση αποστολής δεδομένων, ο HC διαβάζει τα δεδομένα από τη διεύθυνση της κύριας μνήμης που του υποδεικνύει το HCD και τα στέλνει στη συσκευή στόχο. Σε περίπτωση λήψης δεδομένων ο HC λαμβάνει τα δεδομένα από τη συσκευή που τα στέλνει και τα αποθηκεύει στην κύρια μνήμα, στη διεύθυνση που του υποδεικνύει το HCD. Ο HC δημιουργεί μόνο του τα απαραίτητα πακέτα για την εφαρμογή του πρωτοκόλλου USB, ανάλογα με τα δεδομένα που του παρέχει το USBD.

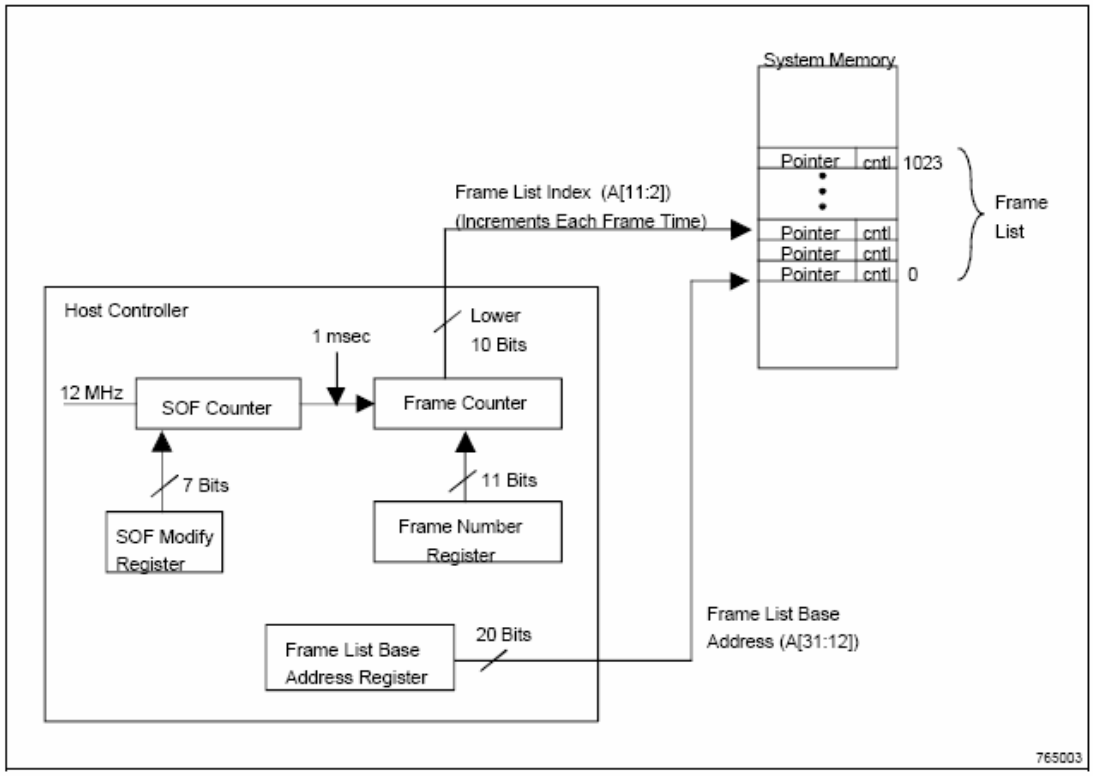
4.1.2 Δομές Δεδομένων και Βασικές Αρχές Χρονοδρομολόγησης του UHCI

Το σχήμα 4.1.2 δείχνει τα βασικά στοιχεία της λειτουργίας του Host Controller. Όπως έχει αναφερθεί στο Κεφάλαιο 2, σε κάθε σύστημα USB ο πραγματικός χρόνος διαρείται σε πλαίσια *(Frames)*, διάρκειας 1 ms, μέσα στα οποία γίνονται μεταφορές δεδομένων στον δίαυλο, δηλαδή πραγματοποιείται ανταλλαγή πακέτων δεδομένων μεταξύ του USB Host και των συσκευών USB. Η αρχή κάθε frame σηματοδοτείται με την εκπομπή από τον HC ενός πακέτου SOF (*Start-of-frame*) κάθε 1 ms. Με τον τρόπο αυτό ο HC υποστηρίζει μεταφορά δεδομένων πραγματικού χρόνου *(real-time)* όπως θα φανεί παρακάτω. Το πακέτο SOF στέλνεται όταν ο SOF Counter του HC εκπνέει. Η default τιμή του SOF Counter αντιστοιχεί σε χρόνο ενός Frame ίσο με 1 ms. Ο χρόνος του Frame μπορεί να αλλάξει σε μικρό βαθμό, για λόγους συγχρονισμού με το υπόλοιπο σύστημα, προγραμματίζοντας κατάλληλα τον καταχωρητή SOF Modify.

Ο HC τοποθετεί τον αριθμό του frame *(Frame Number*) σε κάθε πακέτο SOF. Μετά το SOF ακολουθούν τα πακέτα, τα οποία εκπέμπονται από τον HC και τις συσκευές, που πραγματοποιούν τα transactions που είναι προς εξυπηρέτηση, και στο

τέλος του frame εκπέμπεται η κατάσταση EOF, πριν η τιμή του frame number αυξηθεί κατά ένα και αρχίσει η επεξεργασία του επόμενου frame.

Όπως έχει αναφερθεί στο Κεφάλαιο 3, το HCD κατασκευάζει μια δομή δεδομένων, η οποία αναφέρεται ως Transaction List, μέσω της οποίας παραδίδει στον HC το χρονοδιάγραμμα (*Schedule*) των αιτήσεων μεταφοράς δεδομένων προς εκτέλεση. Στο πρότυπο UHCI η παραπάνω δομή δεδομένων αναφέρεται ως Frame List και αποτέλει έναν πίνακα 1024 καταχωρήσεων (*entries*), όπου κάθε καταχώρηση αντιστοιχεί σε ένα frame στο USB. Κάθε καταχώρηση περιέχει ένα δείκτη προς το χρονοδιάγραμμα (*Schedule*) του αντίστοιχου frame. Ο HC διατηρεί real-time συγχρονισμό μεταξύ των frames και των δεδομένων, αντιστοιχώντας το frame number με την εκτέλεση της αντίστοιχης καταχώρησης του frame list. Ο Frame Counter παράγει το frame number και το εισάγει σε κάθε SOF πακέτο. Ο frame counter μπορεί να προγραμματιστεί μέσω του Frame Number Register και αυξάνει κατά ένα μετά από κάθε frame.Ο HC χρησιμοποιεί τα κατώτερα 10 bits του frame number σαν δείκτη στα 1024 entries του frame list, το οποίο βρίσκεται στην κύρια μνήμη. Με τον τρόπο αυτό ο HC επεξεργάζεται κάθε entry του frame list κατά τη διάρκεια του αντίστοιχου frame, και μετά το τέλος του frame, επεξεργάζεται το επόμενο entry του frame list. Αυτός ο τρόπος λειτουργίας εγγυάται ότι τα isochronous transfers θα εκτελούνται με σταθερό ρυθμό μεταφοράς δεδομένων, αφού οι αντίστοιχες αιτήσεις έχουν σταθερό μήκος δεδομένων προς μεταφορά (βλέπε Παράγραφο 2.2.6) και εκτελούνται κάθε 1 ms, δηλαδή σε διαδοχικά frames.



Σχήμα 4.1.2: Αντιστοιχία μεταξύ του Frame Number και των entries του Frame List

Συνολικά οι δομές δεδομένων που χρησιμοποιούνται στο UHCI για την κατασκευή του χρονοδιαγράμματος των αιτήσεων μεταφοράς δεδομένων είναι:

Frame List, Transfer Descriptors και Queue Heads. Κάθε καταχώρηση του Frame List δείχνει στο Schedule του αντίστοιχου frame το οποίο δομείται συνδέοντας κατάλληλα τα διάφορα Transfer Descriptors και Queue Heads. Ο Host Controller υποστηρίζει τα τέσσερα είδη μεταφοράς δεδομένων *(transfer types)* που ορίζει το πρωτόκολλο USB: Isochronous, Interrupt, Control και Bulk. Για κάθε τύπο μεταφοράς δεδομένων, η δομή δεδομένων που καθορίζει τα χαρακτηριστικά ενός transaction είναι το Transfer Descriptor *(TD)*. Οι Isochronous transfers απαιτούν σταθερό ρυθμό μετάδοσης *(fixed transfer rate)*, αλλά δεν απαιτούν εγγυημένη παράδοση δεδομένων (έλεγχο λαθών) και δεν γίνεται επανάληψη των αποτυχημένων transactions. Έτσι τα TD που σχετίζονται με τα Isochronous transactions, αποσύρονται από τη μνήμη ανεξάρτητα από το αποτέλεσμά τους.

Αντίθετα, τα υπόλοιπα είδη των transfer types, προϋποθέτουν ένα μηχανισμό εγγυημένης παράδοσης δεδομένων. Έτσι τα TD που αντιστοιχούν σε non-isochronous transfers δεν αποσύρονται από τη μνήμη αν δεν ολοκληρωθούν επιτυχώς τα αντίστοιχα transactions, αλλά παραμένουν σε αυτή και τα transactions επαναλαμβάνονται. Ο αριθμός των επαναλήψεων ενός συγκεκριμένου transaction περιορίζεται από ένα όριο λαθών το οποίο ορίζεται στο αντίστοιχο TD. Σε περίπτωση που το όριο των λαθών ξεπεραστεί, το TD του transaction απενεργοποιείται (τίθεται inactive). Το HCD πρέπει να παρέχει τους απαραίτητους μηχανισμούς για να ξεπεραστεί το πρόβλημα.

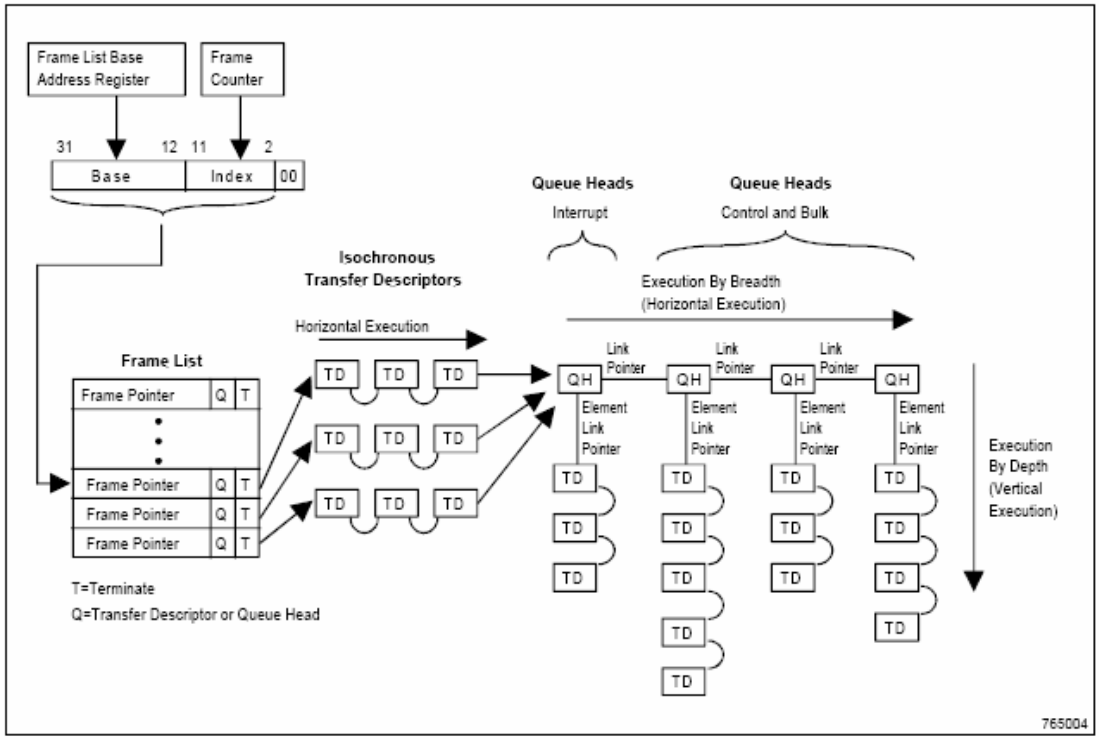
Η ροή μέσα στο Schedule βασίζεται σε δείκτες (*Link Pointers*) που βρίσκονται στο Frame List, τα Transfer Descriptors και τα Queue Heads. Με τα Link Pointers επιτυγχάνεται η σύνδεση ανάμεσα στα στοιχεία (*Data Objects*) του Schedule. Σε ένα Link Pointer καταγράφεται η φυσική διεύθυνση στη μνήμη του επόμενου Data Object του Schedule. Στην αρχή του frame, ο HC διαβάζει τον Link Pointer που θα βρει στο entry του Frame List που αντιστοιχεί στο frame, και ακολουθεί τους Link Pointers που θα βρει στο δρόμο του μέχρι την ολοκλήρωση του Frame ή το τέλος του Schedule. Όποτε ο HC βρίσκει ένα TD το εκτελεί και στη συνέχεια ακολουθεί τον Link Pointer του TD για να βρει το επόμενο Data Object. Παρακάτω περιγράφονται τα βασικά στοιχεία των δομών που χρησιμοποιεί το UHCI.

Frame List: Είναι ένα array από 1024 entries, που το κάθε ένα αντιπροσωπεύει ένα χρονικό πλαίσιο. Κάθε entry συνδέεται με τα data objects που αντιπροσωπεύουν τα transactions που πρέπει να εκτελεστούν στο αντίστοιχο frame μέσω του Link Pointer που περιέχει. Τα πεδία του Frame List μπορούν να διαχειριστούν μόνο από το HCD. Ο HC βρίσκει το entry που αντιστοιχεί στο τρέχον frame συνδυάζοντας την τιμή του Frame List Base Address Register (περιέχει τη διεύθυνση της αρχής του Frame List) και του Frame Counter. Το Frame List είναι μεγέθους 4 Kbytes.

Transfer Descriptors (TD): Ένα TD περιέχει ένα δείκτη για τον καταχωρητή (*buffer*) που θα χρησιμοποιηθεί για τα δεδομένα του transaction (ή NULL αν δε χρησιμοποιούνται δεδομένα) και όλες τις απαραίτητες πληροφορίες που καθορίζουν όλες τις παραμέτρους του transaction. Τα TD χτίζονται από το HCD, αλλά κάποια πεδία τους είναι εγγράψιμα από τον HC, ώστε να επιστραφεί το status του transaction μετά την εκτέλεσή του. Όλα τα TD έχουν την ίδια δομή. Τα TD μπορεί να αντιστοιχούν σε isochronous transfers ή σε non-isochronous transfers ανάλογα με τη θέση τους στο Schedule και την τιμή κάποιων πεδίων ελέγχου τους. Τα isochronous TD τοποθετούνται από το HCD στο σωστό frame σε οριζόντια σειρά, όπως στο

Σχήμα 4.1.3. Το Link Pointer του frame list δείχνει το πρώτο isochronous TD. Το Link Pointer κάθε isochronous TD δείχνει το επόμενο isochronous TD, εκτός από το τελευταίο που δείχνει το πρώτο Queue Head του Schedule (ή δεν δείχνει τίποτα αν τελείωνει το Schedule του frame). Όταν συναντάται ένα TD εκτελείται και γίνεται inactive, ανεξάρτητα από το αποτέλεσμα του transaction.

Queue Heads (QH): Τα QH είναι δομές που οργανώνουν τα non-isochronous TD σε ουρές. Ένα QH και τα TD της ουράς του διαμορφώνουν ένα "Q Context". Τα Interrupt, Control και Bulk transfer types τοποθετούνται σε ουρές, όπως φαίνεται στο Σχήμα 4.1.3. Οι ουρές μπορούν να βρίσκονται στο Schedule αμέσως μετά το Link Pointer του Frame List ή μετά το τελευταίο isochronous TD. Τα QH περιέχουν δύο Link Pointers: Ένα κατακόρυφο pointer που δείχνει το επόμενο TD προς εκτέλεση στο Q Context, και έναν οριζόντιο pointer που δείχνει το επόμενο QH ή TD του Schedule. Επίσης, είναι δυνατόν ο κατακόρυφος δείκτης να δείχνει σε ένα QH. Τα TD που βρίσκονται σε ένα Q Context και δεν ολοκληρώνονται επιτυχώς επανεκτελούνται. Υπάρχουν δύο τρόποι διάσχισης των ουρών: Στη διάσχιση κατά βάθος (Execution by Depth) η εκτέλεση των TD συνεχίζει στο επόμενο TD του Q Context μέχρι το τέλος της ουράς ή κάποιο σφάλμα, και στη συνέχεια εκτελούνται τα TD του επόμενου Q Context. Στη διάσχιση κατά πλάτος (Execution by Breadth) κάθε φορά εκτελείται ένα TD από κάθε Q Context. Ο τρόπος διάσχισης καθορίζεται από κατάλληλο πεδίο του TD.



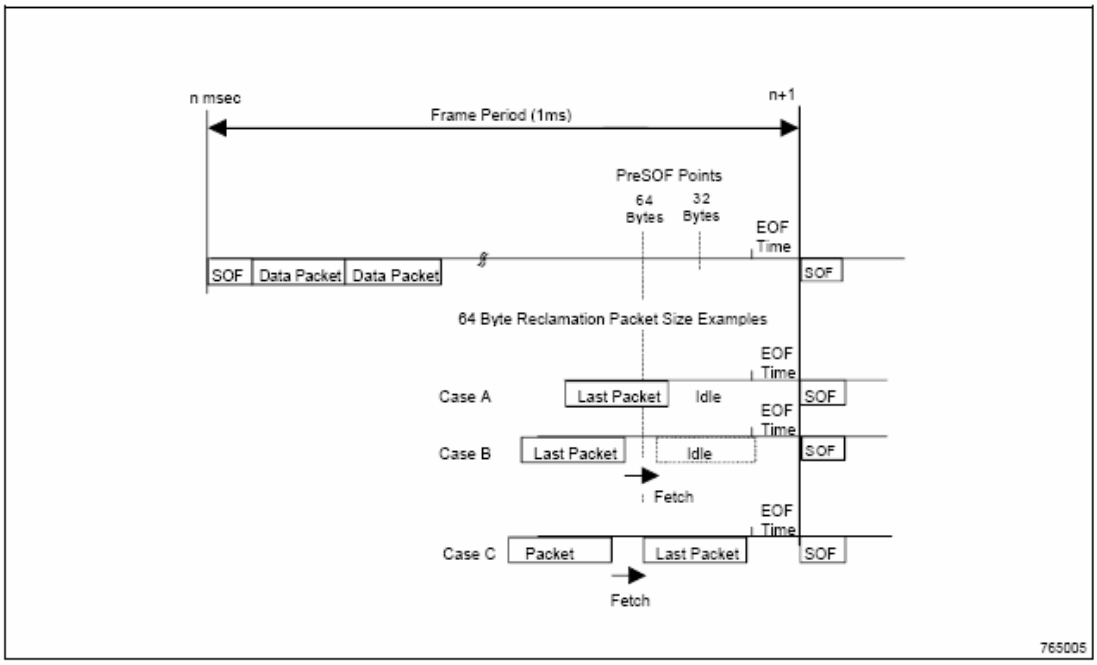
Σχήμα 4.1.3: Μορφή του Schedule

Το HCD κατασκευάζει και τοποθετεί κατάλληλα τις δομές δεδομένων, έτσι ώστε να εγγυηθεί ότι τα isochronous transfers έχουν τη μεγαλύτερη προτεραιότητα. Το Scheduling του HCD επιτρέπει μέχρι το 90% του bandwidth του frame να χρησιμοποιηθεί για isochronous και interrupt transactions, και μέχρι 10% του

bandwidth για control transactions. Το υπολοιπόμενο bandwidth μπορεί να χρησιμοποιηθεί για control και bulk transactions. Έτσι οι διάφορες δομές δεδομένων τοποθετούνται στο schedule με τρόπο όπως αυτόν του παραδείγματος του σχήματος 4.1.3.

4.1.3 Αποκατάσταση του Εύρους Ζώνης

Τα control και bulk transfers τοποθετούνται τελευταία στο schedule για αποκατάσταση του εύρους ζώνης *(Bandwidth Reclamation)*. Το bandwidth reclamation επιτρέπει στο υλικό να συνεχίζει να εκτελεί το schedule μέχρι το τέλος του frame, κάνοντας κύκλους μέσα στα entries των ουρών. Τα control transfers τοποθετούνται πιο μπροστά από τα bulk για να τους δοθεί προτεραιότητα. Το UHCI επιτρέπει το bandwidth reclamation μόνο για full speed control και bulk transfers. Αυτό επιτυγχάνεται θέτοντας το δείκτη του τελευταίου QH να δείχνει το πρώτο full speed control ή bulk transfer. Αν το bandwidth reclamation δεν χρειάζεται, το τελευταίο QH περιέχει ένα bit τερματισμού (terminate bit) και ο HC περιμένει μέχρι το τέλος του frame για να αρχίσει την εκτέλεση του επόμενου frame. Το HCD πρέπει να τοποθετεί τα low speed control transfers στο schedule με τέτοιο τρόπο, έτσι ώστε να ολοκληρωθούν εγγυημένα στο τρέχον frame. Το USB specification δεν επιτρέπει low speed bulk transfers.



Σχήμα 4.1.4: Bandwidth Reclamation

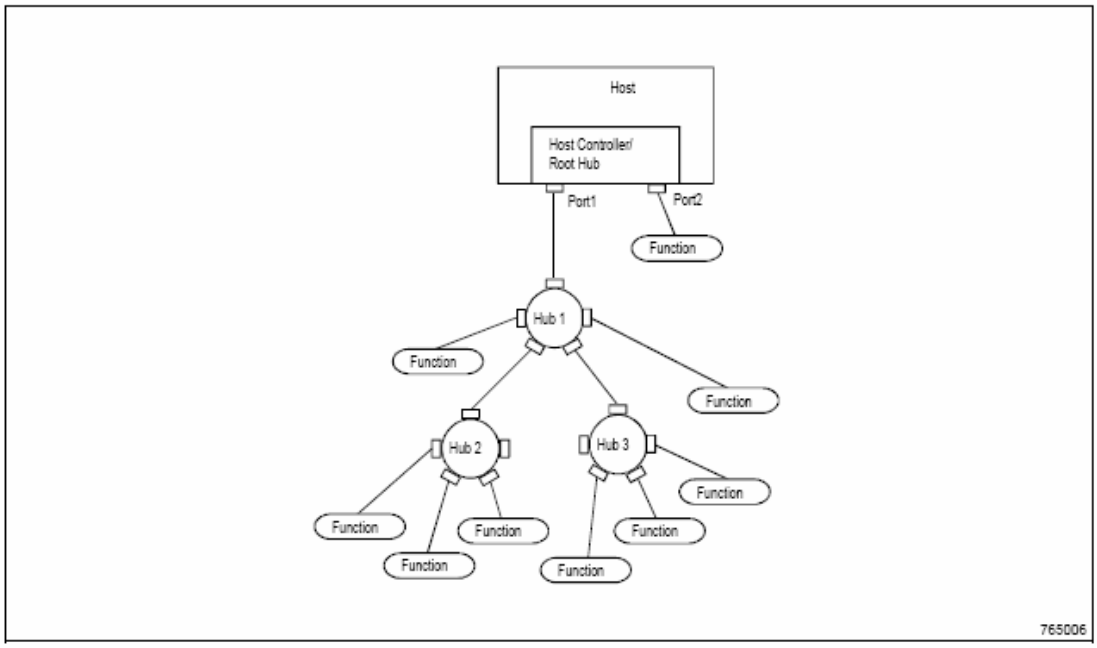
Το HCD επιλέγει το επιθυμητό μέγιστο μέγεθος των πακέτων που θα χρησιμοποιηθεί για bandwidth reclamation, θέτοντας κατάλληλη τιμή στο αντίστοιχο bit του καταχωρητή USB Command. Το μέγεθος αυτό μπορεί να είναι 32 ή 64 bytes. Ο HC χρησιμοποιεί ένα preSOF χρονικό σημείο για να καθορίσει εάν υπάρχει αρκετός χρόνος στο frame για να εκτελεστεί ένα transaction του επιλεγμένου

μεγέθους. Με τον τρόπο αυτό ο HC μεγιστοποιεί τη χρησιμοποίηση του διαύλου, δίνοντας άεργο χρόνο για την πραγματοποίηση επιπλέον transactions. Κανένα πακέτο που πρόκεται να χρησιμοποιηθεί για bandwidth reclamation δεν πρέπει να είναι μεγαλύτερο από το μέγεθος που έχει επιλεγεί στον USB Command Register. Το HCD πρέπει να εγγυάται ότι οποιοδήποτε πακέτο στο schedule του frame έχει μεγαλύτερο μέγεθος από το μέγιστο που χρησιμοποιείται για bandwidth reclamation θα ολοκληρωθεί στο τρέχον frame.

Το σημείο preSOF επίσης εμποδίζει κάθε πακέτο που μπορεί να μην χωράει στο frame να σταλεί. Στο Σχήμα 4.1.4 δείχνονται τρεις χαρακτηριστικές περιπτώσεις: Στην περίπτωση Α (*Case A*), τη χρονική στιγμη του preSOF ο HC δεν έχει ολοκληρώσει τη φάση ενημέρωσης του status του προηγούμενου TD. Ο HC δεν διαβάζει το επόμενο TD από τη μνήμη. Στην περίπτωση Β (*Case B*), στο σημείο preSOF ο HC δεν έχει διαβάσει πλήρως το επόμενο TD ή δεν έχει φτιάξει το πακέτο που αντιστοιχεί στο TD. Το transaction εγκαταλείπεται και δεν γράφονται δεδομένα πίσω στη μνήμη. Στην περίπτωση Γ (*Case C*), το transaction αρχίζει πριν το preSOF και τελικά ολοκληρώνεται, αλλά είναι το τελευταίο transaction του frame.

4.1.4 Κεντρικός Διακλαδωτής

Ο Host Controller του UHCI υλοποιεί το Κεντρικό Hub *(Root Hub)*, όπως απαιτείται από το USB specification. Το root hub είναι ενσωματωμένο στον HC και έχει δύο Ports. Το UHCI επιτρέπει την υλοποίηση επιπλέον ports, μέχρι τον αριθμό που επιτρέπει το USB Specification. Το root hub παρέχει έλεγχο ροής δεδομένων και διαχείριση ισχύος στις ports. Η κατάσταση των ports του root hub μπορεί να ελεγχθεί από το HCD διαβάζοντας τον καταχωρητή PORTSC κάθε θύρα *(Port)*.



Σχήμα 4.1.5: Τοπολογία των Hubs

4.2 Καταχωρητές του UHCI

Ο Host Controller διαθέτει ένα σύνολο από καταχωρητές για την επικοινωνία με την CPU του υπολογιστικού συστήματος. Χρησιμοποιούνται για λειτουργίες οι οποίες δεν πραγματοποιούνται με αποδοτικό τρόπο μέσω της κύριας μνήμης. Το HCD επεμβαίνει στους καταχωρητές του HC για ρυθμίσει και να δώσει συγκεκριμένες εντολές για εκτέλεση στον HC, αλλά και να ελέγξει το status της τελευταίας transaction που εκτελέστηκε από το υλικό. Οι καταχωρητές αυτοί διακρίνονται σε δύο κατηγορίες: I/O registers και PCI Configuration Registers. Η δεύτερη κατηγορία καταχωρητών απαντάται μόνο σε συσκευές PCI που υλοποιούν έναν Host Controller.

Οι USB Host Controller I/O Registers βρίσκονται στο I/O Space του υπολογιστικού συστήματος και ελέγχουν τις διάφορες λειτουργίες του Host Controller. Η διεύθυνση βάσης *(Base Address)* των καταχωρητών αυτών παρέχεται από έναν PCI Configuration Register, τον USBBASE Register. Στον Πίνακα 4.1 παρουσιάζονται συνοπτικά οι καταχωρητές αυτής της κατηγορίας.

Πίνακας 4.2.1: USB I/O Registers

I/O Address	Mnemonic	Register Description	Register Access
Base Address + (00-01h)	USBCMD	USB Command	R/W
Base Address + (02-03h)	USBSTS	USB Status	R/WC
Base Address + (04-05h)	USBINTR	USB Interrupt Enable	R/W
Base Address + (06-07h)	FRNUM	Frame Number	R/W**
Base Address + (08-0Bh)	FRBASEADD	Frame List Base Address	R/W
Base Address + 0Ch	SOFMOD	Start Of Frame Modify	R/W
Base Address + (10-01h)	PORTSC1	Port 1 Status/Control	R/WC**
Base Address + (00-01h)	PORTSC2	Port 2 Status/Control	R/WC**

Οι PCI Configuration Registers βρίσκονται στο PCI Configuration Space του υπολογιστικού συστήματος και είναι απαραίτητοι για την υποστήριξη ενός PCI USB Host Controller. Χρησιμοποιούνται για την αναγνώριση της συσκευής από τον Απαριθμητή PCI (*PCI Enumerator*), ώστε να δωθεί στην συσκευή ο απαραίτητος χώρος στην μνήμη Εισόδου-Εξόδου (*I/O Space*). Δεν εκτελούν κάποια άλλη λειτουργία. Στον Πίνακα 4.2 παρουσιάζονται συνοπτικά οι καταχωρητές αυτής της κατηγορίας.

Πίνακας 4.2.2: PCI Configuration Registers

Configuration Offset	Mnemonic	Register Description	Register Access
09-0Bh	CLASSC	Class Code	RO
20-23h	USBBASE	I/O Space Base Address	R/W

60h	SBRN	Serial Bus Release Number	RO

Σημειώνεται ότι εκτός από τους καταχωρητές που παρουσιάζονται στον παραπάνω πίνακα, υπάρχουν και άλλοι καταχωρητές στο PCI Configuration Offset, η χρησιμότητα των οποίων εξαρτάται από την υλοποίηση της PCI συσκευής. Στον πίνακα 4.2.3 εξηγείται ο συμβολισμός που χρησιμοποιείται για τη δυνατότητα πρόσβασης του λογισμικού στους καταχωρητές. Ακολουθεί η παρουσίαση των παραπάνω καταχωρητών. Η παρουσίαση περιορίζεται στη μορφή των καταχωρητών και σε μια γενική περιγραφή της λειτουργίας τους. Στο Παράρτημα Α παρουσιάζεται αναλυτικά η λειτουργία των bits των καταχωρητών του UHCI..

Πίνακας 4.2.3: HC Registers Access

RO	**Read Only.** Το γράψιμο σε αυτούς τους καταχωρητές αγνοείται.
WO	**Write Only.** Το διάβασμα σε αυτούς τους καταχωρητές δεν έχει αποτέλεσμα
R/W	**Read/Write.** Οι καταχωρητές αυτοί μπορούν να διαβαστούν ή να γραφτούν. Σε μερικούς καταχωρητές κάποια συγκεκριμένα bits είναι Read Only.
R/WC	**Read/Write Clear.** Τα bits αυτών των καταχωρητών μπορούν να διαβαστούν ή να γραφτούν. Όμως, γράφοντας 1 σε ένα bit το θέτει 0 (clear) ενώ το γράψιμο ενός 0 δεν αλλάζει το αντίστοιχο bit.
**	Οι καταχωρητές αυτοί μπορούν να γραφτούν μόνο ως WORD. Το γράψιμο μεμονωμένων bytes σε αυτούς τους καταχωρητές δίνει απρόβλεπτα αποτελέσματα.

4.2.1 USB I/O Registers

4.2.1.1 USBCMD – USB Command Register

Πίνακας 4.2.4: Χαρακτηριστικά του USB Command Register

I/O Address	Base Address + (00-01h)
Default Τιμή	0000h
Access	Read/Write
Μέγεθος	16 bits

Ο καταχωρητής USBCMD περιέχει την εντολή που πρόκειται να εκτελεστεί από τον HC. Το γράψιμο μιας εντολής στον καταχωρητή αυτό έχει ως αποτέλεσμα την εκτέλεση της εντολής.

Το λογισμικό μπορεί να δώσει εντολή στον HC να αρχίσει ή να σταματήσει την εκτέλεση του Schedule, θέτοντας κατάλληλα τα bits του καταχωρητή. Επίσης μπορεί να επιλεγεί ο τρόπος *(mode)* λειτουργίας του HC: Normal mode ή Debug Mode. Στο Debug mode ο HC εκτελεί το επόμενο Transfer Descriptor του Schedule και σταματά.

Όταν ο HC τεθεί και πάλι σε λειτουργία, συνεχίζει από το σημείο του Schedule όπου σταμάτησε. Στο Normal mode ο HC εκτελεί το Schedule χωρίς παύσεις, εκτός εάν λάβει εντολή να σταματήσει ή συμβεί κάποιο σφάλμα. Μέσω αυτού του καταχωρητή μπορεί να γίνει επανεκίνηση *(Reset)* του HC ή όλου του συστήματος USB. Τέλος, στον USBCMD επιλέγεται το μέγιστο μέγεθος των πακέτων που χρησιμοποιούνται για Bandwidth Reclamation.

Πίνακας 4.2.5: USB Command Register

Bit	Description
15:8	**Reserved**
7	**Max Packet**
6	**Configure Flag (CF)**
5	**Software Debug (SWDBG)**
4	**Force Global Resume (FGR)**
3	**Enter Global Suspend Mode (EGSM)**
2	**Global Reset (GRESET)**
1	**Host Controller Reset (HCRESET)**
0	**Run/Stop (RS)**

4.2.1.2 USBSTS – USB Status Register

Πίνακας 4.2.6: Χαρακτηριστικά του USB Status Register

I/O Address	Base Address + (02-03h)
Default Τιμή	0000h
Access	Read/Write Clear
Μέγεθος	16 bits

Ο καταχωρητής USBSTS υποδεικνύει διακοπές που εκκρεμούν, καταστάσεις σφαλμάτων του HC, και άλλες καταστάσεις λειτουργίας του HC. Δεν εμφανίζει το status του τελευταίου transaction στο USB. Το λογισμικό θέτει 0 ένα bit του καταχωρητή, γράφοντας 1 στη θέση του bit.

Πίνακας 4.2.7: USB Status Register

Bit	Description
15:6	**Reserved**
5	**HCHalted**
4	**Host Controller Process Error**
3	**Host System Error**
2	**Resume Detect**
1	**USB Error Interrupt**
0	**USB Interrupt (USBINT)**

4.2.1.3 USBINTR – USB Interrupt Enable Register

Πίνακας 4.2.8: Χαρακτηριστικά του USB Interrupt Enable Register

I/O Address	Base Address + (04-05h)
Default Τιμή	0000h
Access	Read/Write
Μέγεθος	16 bits

Ο καταχωρητής USBINTR ενεργοποιεί *(Enable)* ή απενεργοποιεί *(Disable)* την αναφορά συγκεκριμένων διακοπών *(Interrupts)* στο λογισμικό. Όταν ένα bit του καταχωρητή αυτού είναι 1 και η αντίστοιχη διακοπή είναι ενεργή, δημιουργείται διακοπή στον Host. Κρίσιμα σφάλμα *(Fatal Errors)*, όπως το Host Controller Processor Error δεν μπορούν να απενεργοποιηθούν από τον HC. Τα αποτελέσματα των διαφόρων διακοπών που απενεργοποιούνται σε αυτόν τον καταχωρητή εξακολουθούν να εμφανίζονται στον Status Register, ώστε να δίνεται η δυνατότητα στο HCD να κάνει polling για τις διάφορες καταστάσεις που προέκυψαν κατά τη λειτουργία του.

Πίνακας 4.2.9: USB Interrupt Enable Register

Bit	Description
15:4	**Reserved**
3	**Short Packet Interrupt Enable**
2	**Interrupt On Complete (IOC) Enable**
1	**Resume Interrupt Enable**
0	**Timeout/CRC Interrupt Enable**

4.2.1.4 FRNUM – Frame Number Register

Πίνακας 4.2.10: Χαρακτηριστικά του Frame Number Register

I/O Address	Base Address + (06-07h)
Default Τιμή	0000h
Access	Read/Write (Word Writes)
Μέγεθος	16 bits

Ο καταχωρητής FRNUM περιέχει το τρέχον frame number το οποίο περιέχεται στο πακέτο SOF του frame. Το περιεχόμενο του καταχωρητή αντιστοιχεί στην τιμή του εσωτερικού Frame Number Counter του HC και χρησιμοποιείται για να επιλεγεί ένα συγκεκριμένο entry στο Frame List κατά την εκτέλεση του Schedule. Ο καταχωρητής ενημερώνεται στο τέλος του χρόνου κάθε frame. Ο καταχωρητής αυτός πρέπει να γράφεται σαν word (και τα 16 bits). Οι εγγραφές bytes δεν υποστηρίζονται.

Πίνακας 4.2.11: Frame Number Register

Bit	Description
15:11	**Reserved**
10:0	**Frame List Current Index/Frame Number**

4.2.1.5 FRBASEADD – Frame List Base Address Register

Πίνακας 4.2.12: Χαρακτηριστικά του Frame List Base Address Register

I/O Address	Base Address + (08-0Bh)
Default Τιμή	Μη καθορισμένη
Access	Read/Write
Μέγεθος	32 bits

Ο 32-bit καταχωρητής FRBASEADD περιέχει τη διεύθυνση στην κύρια μνήμη του υπολογιστικού συστήματος στην οποία αρχίζει το Frame List. Το HCD φορτώνει τον καταχωρητή αυτό πριν από την εκτέλεση του Schedule από τον HC. Όταν ο καταχωρητής γράφεται, χρησιμοποιούνται μόνο τα 20 πιο σημαντικά bits. Τα 12 κατώτερα bits γράφονται ως μηδέν, δίνοντας 4-Kbyte alignment. Τα περιεχόμενα του FRBASEADD συνδυάζονται με τον Frame Number Counter έτσι ώστε ο HC να διατρέξει διαδοχικά τα entries του Frame List. Τα δύο λιγότερο σημαντικά bits είναι πάντα 00. Αυτό προυποθέτει DWord alignment για όλα τα entries του Frame List. Με αυτόν τον τρόπο προκύπτουν 1024 entries στο Frame List.

Πίνακας 4.2.13: Frame List Base Address Register

Bit	Description
31:12	**Base address**
11:0	**Reserved**

4.2.1.6 SOFMOD – Start of Frame Modify Register

Πίνακας 4.2.14: Χαρακτηριστικά του Start Of Frame Modify Register

I/O Address	Base Address + (0Ch)
Default Τιμή	40h
Access	Read/Write
Μέγεθος	8 bits

Ο 8-bit καταχωρητής SOFMOD χρησιμοποιείται για την ρύθμιση του χρονισμού της παραγωγής των SOF πακέτων στο USB. Χρησιμοποιούνται μόνο τα 7

λιγότερο σημαντικά bits. Κάθε τιμή του καταχωρητή SOFMOD αντιστοιχεί σε ένα διαφορετικό Frame Time. Όταν η τιμή του καταχωρητή αλλάζει μετά από εγγραφή από το λογισμικό, ο χρονισμός των SOF πακέτων θα προσαρμοστεί ανάλογα από το επόμενο frame. Χρησιμοποιώντας αυτόν τον καταχωρητή, ο χρόνος ενός frame μπορεί να μεταβληθεί σε μεγάλο βαθμό καλύπτοντας τις απαιτήσεις του USB specification. Επίσης, ο SOFMOD register μπορεί να χρησιμοποιηθεί για τη διατήρηση real-time συγχρονισμού με το υπόλοιπο σύστημα, έτσι ώστε όλες οι συσκευές να έχουν την ίδια αίσθηση του πραγματικού χρόνου. Η αρχική τιμή του καταχωρητή εξαρτάται από το σύστημα και βασίζεται στην ακρίβεια του ρολογιού του USB. Αρχικοποιείται από το BIOS του συστήματος. Ο καταχωρητής μπορεί να γραφτεί οποιαδήποτε στιγμή.

Πίνακας 4.2.15: Start Of Frame Modify Register

Bit	Description
7	**Reserved**
6:0	**SOF Timing Value**

4.2.1.7 PORTSC – Port Status and Control Register

Πίνακας 4.2.16: Χαρακτηριστικά του Port Status and Control Register

I/O Address	Base Address + (10-11h) – Port 1 Base Address + (12-13h) – Port 2
Default Τιμή	0080h
Access	Read/Write (Word)
Μέγεθος	16 bits

Οι καταχωρητές PORTSC χρησιμοποιούνται για να διαβαστεί η κατάσταση των ports του Root Hub. Έτσι το λογισμικό μπορεί να διαβάσει αν είναι συνδεδεμένη μια συσκευή σε ένα port, αν η συσκευή είναι low-speed ή full-speed, εάν υπήρξε αλλαγή στη συνδεσιμότητα του port, αν το port είναι ενεργοποιημένο, και άλλες σχετικές πληροφορίες.

Πίνακας 4.2.17: Port Status and Control Register

Bit	Description
15:13	**Reserved**
12	**Suspend – R/W**
11:10	**Reserved**
9	**Port Reset – R/W**
8	**Low Speed Device Attached – RO**
7	**Reserved-RO**
6	**Resume Detect – R/W**
5:4	**Line Status – RO**

3	**Port Enable/Disable Change – R/WC**
2	**Port Enabled/Disabled – R/W**
1	**Connect Status Change – R/WC**
0	**Current Connect Status – RO**

4.2.2 PCI Configuration Registers

4.2.2.1 CLASSC – Class Code Register

Πίνακας 4.2.18: Χαρακτηριστικά του Class Code Register

Address Offset	09-0Bh
Default Τιμή	010180h
Access	Read Only
Μέγεθος	24 bits

Ο καταχωρητής CLASSC περιέχει τις πληροφορίες για το programming interface που σχετίζονται με το Sub-Class Code και το Base Class Code της συσκευής. Επίσης περιέχει τις πληροφορίες για το Base Class Code και το αντίστοιχο Sub-Class Code. Έτσι, ο καταχωρητής CLASSC περιέχει την ταυτότητα της PCI συσκευής.

Πίνακας 4.2.19: Class Code Register

Bit	Description
23:16	**Base Class Code (BASEC)**
15:8	**Sub-Class Code (SCC)**
7:0	**Programming Interface (PI)**

4.2.2.2 USBBASE – I/O Space Base Address Register

Πίνακας 4.2.20: Χαρακτηριστικά του I/O Space Base Address Register

Address Offset	20-23h
Default Τιμή	00000001h
Access	Read/Write
Μέγεθος	32 bits

Ο καταχωρητής USBBASE περιέχει τη διεύθυνση βάσης *(Base Address)* των USB I/O Registers.

Πίνακας 4.2.21: I/O Space Base Address Register

Bit	Description
31:16	**Reserved**
15:5	**Index Register Base Address**
4:1	**Reserved**
0	**Resource Type Indicator (RTE)-RO**

4.2.2.3 SBRN - Serial Bus Release Number Register

Πίνακας 4.2.22: Χαρακτηριστικά του Serial Bus Release Number Register

Address Offset	60h
Default Τιμή	Ανάλογα με τη συσκευή
Access	Read Only
Μέγεθος	8 bits

Ο καταχωρητής SBRN περιέχει την έκδοση του Universal Serial Bus Specification με την οποία είναι συμβατός ο Host Controller που υλοποιεί η συσκευή.

Πίνακας 4.2.23: Serial Bus Release Number Register

Bit	Description
7:0	**Serial Bus Specification Number**

4.3 Δομές Δεδομένων του UHCI

Σε αυτήν την ενότητα περιγράφονται λεπτομερώς οι δομές δεδομένων που χρησιμοποιούνται για την επικοινωνία μεταξύ του HCD και του HC. Οι δομές αυτές ανταλλάσουν δεδομένα για μετάδοση στο USB *(data)*, δεδομένα ελέγχου *(control)*, και δεδομένα κατάστασης *(status)* μεταξύ των δύο μερών του UHCI και βρίσκονται στην κύρια μνήμη του συστήματος. Το HCD πρέπει να δημιουργήσει αυτές τις δομές δεδομένων σύμφωνα με τις προδιαγραφές του UHCI. Τα Frame Lists πρέπει να είναι στοιχισμένα (*aligned*) σε όρια 4 Kbyte (1024 entries μεγέθους 4 bytes). Τα Transfers Descriptors *(TD)* και τα Queue Heads *(QH)* πρέπει να είναι aligned σε όρια 16 bytes (4 DWords).

4.3.1 Frame List Pointer

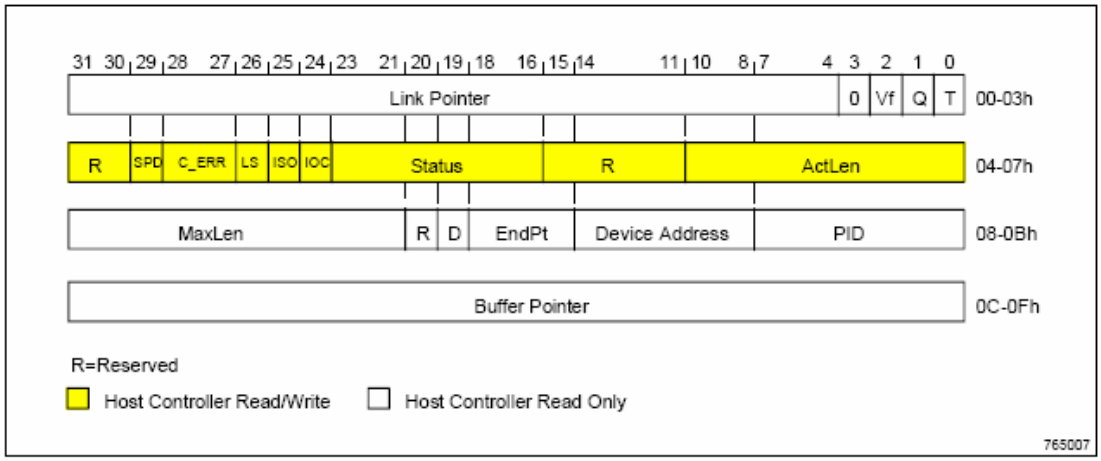
Τα Frame List Pointers *(FLP)* αποτελούν τις καταχωρήσεις (*entries*) των Frame Lists. Κάθε HC χρησιμοποιεί ένα Frame List, το οποίο είναι ένας πίνακας (*array*) από 1024 Frame List Pointers. Τα Frame List Pointers είναι στοιχισμένα (*aligned*) σε όρια ενός DWord (4 bytes) και υπάρχει ένα FLP για κάθε Frame. Περιέχουν ένα Link Pointer (δείκτη διεύθυνσης) προς το πρώτο Data Object (TD ή QH) που θα επεξεργαστεί στο αντίστοιχο Frame. Το Frame List Pointer δηλώνει σαφώς εάν το επόμενο Data Object είναι TD ή QH, ή εάν το Frame είναι κενό.

Πίνακας 4.3.1: Frame List Pointer

Bit	Description
31:4	**Frame List Pointer (FLP)**
3:2	**Reserved**
1	**QH/TD Select**
0	**Terminate (T)**

4.3.2 Transfer Descriptor

Τα Transfer Descriptors *(TD)* περιέχουν τις απαραίτητες πληροφορίες για τα transactions στο USB που αιτήθηκαν από ένα Client στον Host. Είναι aligned σε όρια 16 bytes. Αν και υπάρχουν τέσσερα διαφορετικά είδη μεταφοράς δεδομένων, όλα τα TD έχουν την ίδια μορφή και το είδος της μεταφοράς καθορίζονται από ορισμένα bits TD.Τα TD έχουν μέγεθος 32 byte. Τα τέσσερα τελευταία DWords χρησιμοποιούνται μόνο από το λογισμικό. Τα τέσσερα πρώτα DWords παρουσιάζονται αναλυτικά παρακάτω. Κατά την επεξεργασία τους, ο HC πραγματοποιεί Consistency Checks (ελέγχους ορθότητας) σε κάποια πεδία τους, και αν αποτύχουν, ο HC σταματάει αμέσως και στέλνει μια διακοπή στο σύστημα (USBSTS, HC Process Error).



Σχήμα 4.3.1: Transfer Descriptor

4.3.2.1 TD Link Pointer

Το TD Link Pointer είναι το πρώτο DWord του TD (00-03h) και περιέχει έναν δείκτη προς το επόμενο Data Object και control bits, τα οποία υποδεικνύουν αν ο δείκτης δείχνει σε έγκυρο Data Object και αν αυτό είναι TD ή QH.

Πίνακας 4.3.2: TD Link Pointer

Bit	Description
31:4	**Link Pointer (LP)**
3	**Reserved**
2	**Depth/Breadth Select (Vf)**
1	**QH/TD Select (Q)**
0	**Terminate (T)**

4.3.2.2 TD Control and Status

Το TD Control and Status είναι το δεύτερο DWord του TD (04-07h). Περιέχει πεδία που καθορίζουν το είδος και τα χαρακτηριστικά του transaction που αντιπροσωπεύει το TD. Ο HC ενημερώνει κάποια πεδία του μετά την εκτέλεση του TD για να επιστρέψει το status του transaction.

Πίνακας 4.3.3: TD Control and Status

Bit	Description
31:30	**Reserved (R).**
29	**Short Packet Detect (SPD)**
28:27	**C_Err**
26	**Low Speed Device (LS)**

25	**Isochronous Select (IOS)**
24	**Interrupt on Complete (IOC)**
23:16	**Status**
15:11	**Reserved (R)**
10:0	**Actual Length (ActLen)**

4.3.2.3 TD Token

Το TD Token είναι το τρίτο DWord του TD (08-0Bh). Περιέχει όλες τις πληροφορίες για να δημιουργηθεί το Token του transaction.

Πίνακας 4.3.4: TD Token

Bit	Description
31:21	**Maximum Length (MaxLen)**
20	**Reserved (R).**
19	**Data Toggle (D)**
18:15	**Endpoint (EndPt)**
14:8	**Device Address**
7:0	**Packet Identification (PID)**

4.3.2.4 TD Buffer Pointer

Το TD Buffer Pointer είναι το τέταρτο DWord του TD (0C-0Fh). Περιέχει τη διεύθυνση του buffer που θα χρησιμοποιηθεί για το transaction. Το μέγεθος του buffer πρέπει να είναι μεγαλύτερο ή ίσο από την τιμή του πεδίου Max Length του Dword Token.

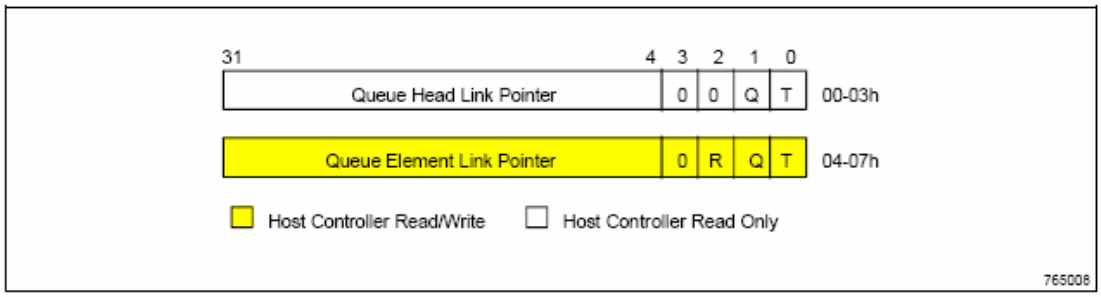
Πίνακας 4.3.5: TD Buffer Pointer

Bit	Description
31:0	**Buffer Pointer**

Τα τελευταία τέσσερα DWords του TD είναι για χρήση αποκλειστικά από το HCD. Το περιεχόμενο τους μπορεί να επιλεγεί ελεύθερα από τον προγραμματιστή.

4.3.3 Queue Head

Τα Queue Heads *(QH)* είναι ειδικές δομές που χρησιμοποιούνται για την υποστήριξη των απαιτήσεων των Control, Bulk και Interrupt transfers. Οι τύποι αυτοί μεταφοράς δεδομένων, σε αντίθεση με τα Isochronous transfers, απαιτούν έλεγχο λαθών και επαναμετάδοση των λανθασμένων πακέτων. Αφού τα TD δεν αποσύρονται αυτόματα μετά την επεξεργασία τους, οι απαιτήσεις τους για απόσυρση ή διατήρηση στο Schedule μπορούν να μειωθούν, βάζοντάς τα σε ουρές. Τα Queue Head αποτελούνται από δύο DWord.



Σχήμα 4.3.2: Queue Head

4.3.3.1 Queue Head Link Pointer

Το Queue Head Link Pointer είναι το πρώτο DWord του QH (00-03h). Περιέχει ένα δείκτη προς το επόμενο Data Object (QH ή TD), μετά την εκτέλεση όλων των απαιτούμενων επεξεργασιών στην ουρά, της οποίας κεφαλή είναι το συγκεκριμένο QH, καθώς και bits ελέγχου, τα οποία δείχνουν αν το επόμενο Data Object είναι έγκυρο και αν είναι TD ή QH.

Πίνακας 4.3.6: Queue Head Link Pointer

Bit	Description
31:4	**Queue Head Link Pointer (QHLP)**
3:2	**Reserved**
1	**QH/TD Select (Q)**
0	**Terminate (T)**

4.3.3.2 Queue Element Link Pointer

Το Queue Element Link Pointer είναι το δεύτερο DWord του QH (04-07h). Περιέχει ένα δείκτη προς το πρώτο Data Object (QH ή TD) της ουράς της οποίας κεφαλή είναι το QH, καθώς και bits ελέγχου, τα οποία δείχνουν αν το επόμενο Data Object είναι έγκυρο και αν είναι TD ή QH.

Πίνακας 4.3.7: Queue Element Link Pointer

Bit	Description
31:4	**Queue Element Link Pointer (QELP)**
3	**Reserved**
2	**Reserved**
1	**QH/TD Select (Q)**
0	**Terminate (T)**

4.4 Χρονοδρομολόγηση στο UHCI

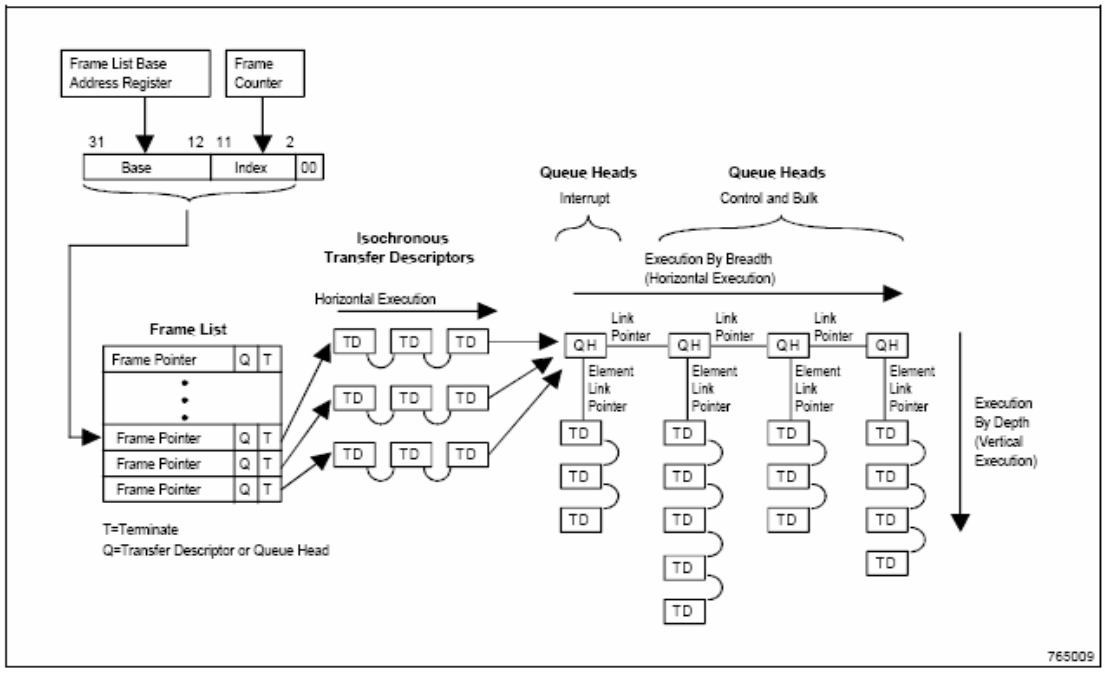
Στην ενότητα 4.1 περιγράφτηκαν οι βασικές αρχές της χρονοδρομολόγησης *(Scheduling)* του UHCI, όπως η προτεραιότητα των αιτήσεων ανάλογα με το είδος της μεταφοράς, η δόμηση του Schedule σε ουρές για τις αιτήσεις interrupt, control και bulk transfers, ο τρόπος διάσχισης του Schedule κατά τη διάρκεια της εκτέλεσης, και η μέθοδος της αποκατάστασης του bandwidth (bandwidth reclamation).

Στην παρούσα ενότητα δίνονται λεπτομέρειες για το πως το HCD και ο HC επικοινωνούν μέσω των δομών δεδομένων του Schedule και δίνονται οι αντίστοιχοι αλγόριθμοι για την εκτέλεση των TD και τη διάσχιση του Schedule.

4.4.1 Εκτέλεση του Χρονοδιαγράμματος (Schedule)

Το HCD δίνει στον HC την αρχική διεύθυνση του Frame List και το index της πρώτης καταχώρησης (*entry*) για εκτέλεση. Στην συνέχεια θέτει 1 το Run/Stop bit του USBCMD για να ξεκινήσει ο HC την εκτέλεση του Schedule. Ο HC επεργάζεται κάθε δομή του Schedule όταν τη συναντά ακολουθώντας τους link pointers.

Η εκτέλεση του Schedule γίνεται όπως φαίνεται το σχήμα 4.4.1. Ο HC διαβάζει ένα entry από το frame list και ελέγχει αν ο δείκτης που περιέχει το entry είναι έγκυρος και σε τι δομή (TD ή QH) αναφέρεται, με βάση τα bit ελέγχου (bits Q και T του σχήματος). Αν υπάρχουν isochcronous transfers στο frame, τότε το link pointer του entry θα δείχνει σε TD, αλλιώς θα δείχνει σε QH ή θα είναι μη έγκυρο, σε περίπτωση που το frame είναι κενό.



Σχήμα 4.4.1: Εκτέλεση του Schedule

Αν το entry του frame list δείχνει σε TD, ο HC διαβάζει το TD και κάνει τις κατάλληλες επεξεργασίες για τη δημιουργία του transaction. Στη συνέχεια ελέγχει το link pointer του TD και διαβάζει το επόμενο data object του schedule, εάν το link pointer είναι έγκυρο. Αν το entry του frame list δείχνει σε QH, ο HC διαβάζει το QH για να καθορίσει τη διεύθυνση του επόμενου data object που πρέπει να εκτελέσει.

Ο HC διαβάζει το επόμενο entry του frame list όταν ο χρόνος του frame εκπνεύσει. Αν ο HC δεν μπορέσει να εκτελέσει όλα τα isochronous TD του frame, αυτά αποσύρονται από το Schedule χωρίς να εκτελεστούν. Υπό κανονικές συνθήκες, αυτό δεν συμβαίνει, αφού τα isochronous TD τοποθετούνται στο Schedule έτσι ώστε να εκτελεστούν όλα. Αν όμως συμβούν διάφορα σφάλματα, όπως babble error, η εκτέλεσή τους θα καθυστερήσει και ίσως κάποια να μην προλάβουν εκτελεστούν.

4.4.1.1 Επεξεργασία των Transfer Descriptors

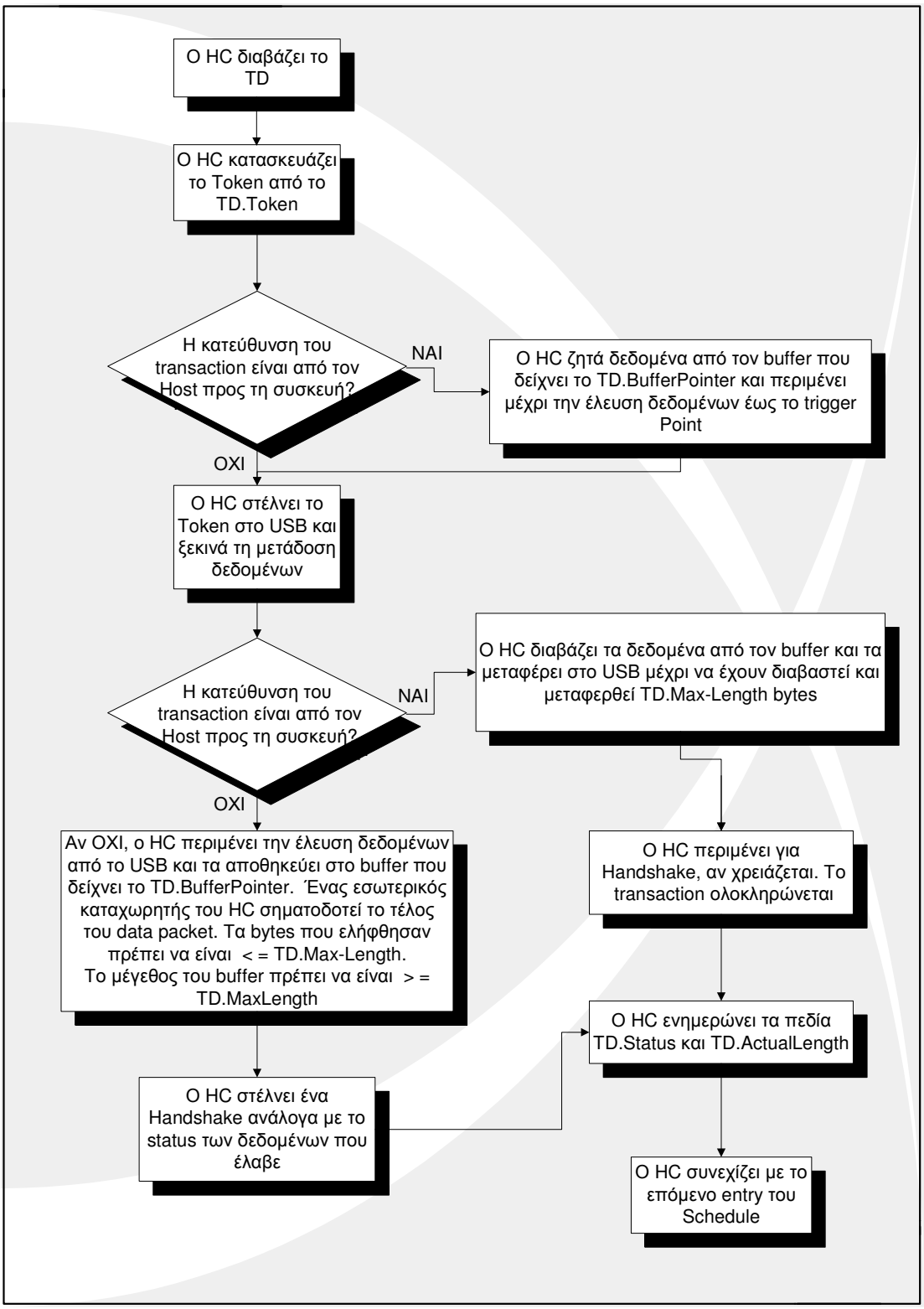
Η επεξεργασία των TD γίνεται σε τρία βήματα:

1. Ο HC διαβάζει και αποκωδικοποιεί το TD. Σε ένα transaction από τον Host προς τη συσκευή, ο HC καθυστερεί το transaction στο USB μέχρι το buffer που συνδέεται με το transaction γεμίσει μέχρι ένα “trigger point”. Όταν αυτό συμβεί ο HC στέλνει το token (βλέπε Ενότητα 2.3) του transaction στο USB. Αν η κατεύθυνση του transaction είναι από τη συσκευή προς τον Host, ο HC στέλνει το token μόλις λάβει και αποκωδικοποιήσει το TD.

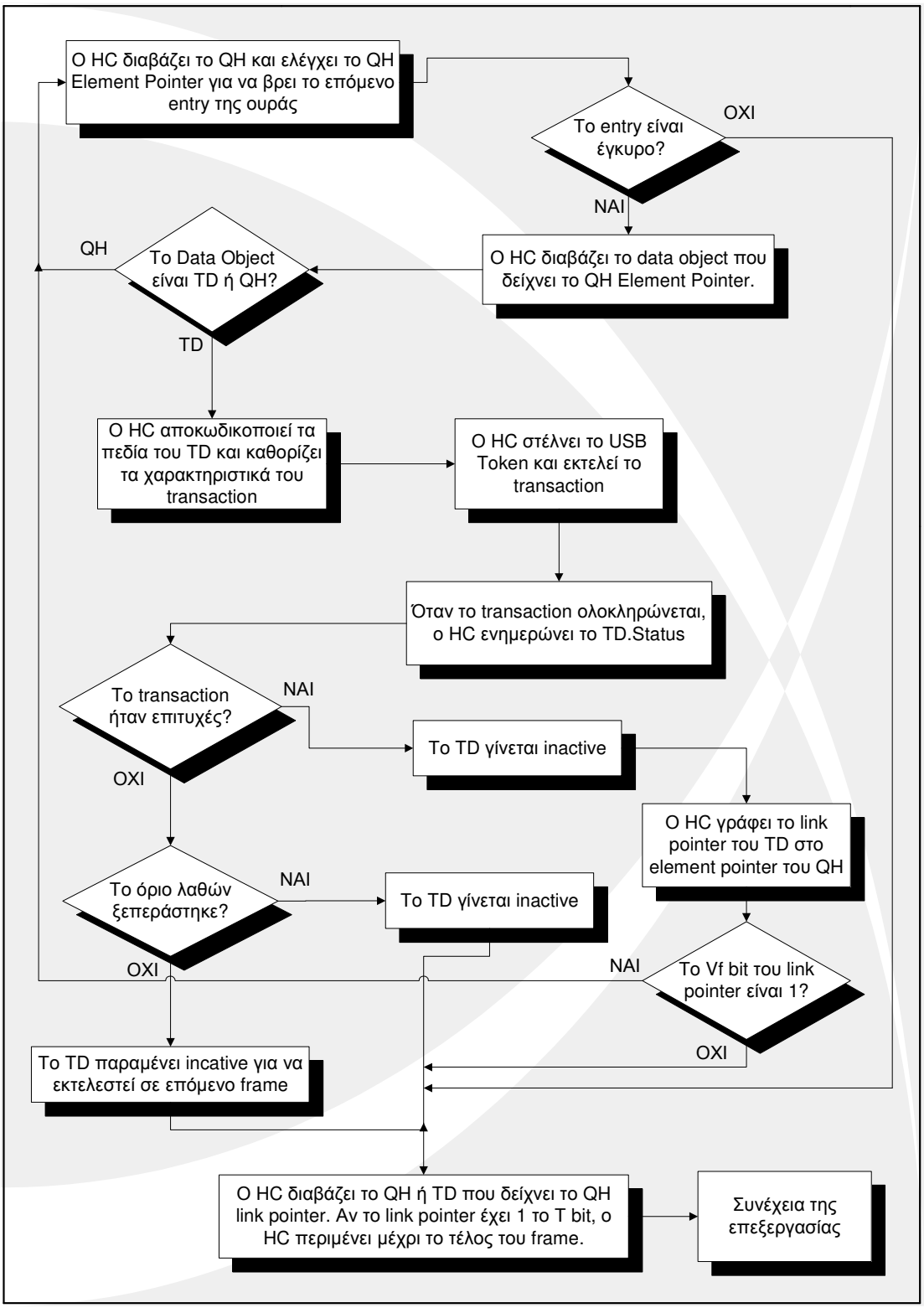
2. Το δεύτερο βήμα είναι το transaction. Ένα transaction χρειάζεται το πολύ τρεις φάσεις για να ολοκληρωθεί. Η διάρκειά του εξαρτάται από την τιμή του πεδιού MaxLen του TD.

3. Το τελευταίο βήμα είναι η ενημέρωση (*update*) των δομών δεδομένων για επεξεργασία μετά την ολοκλήρωση του transaction.

Τα βήματα αυτά είναι κοινά για κάθε είδος TD. Ακολουθούν τρεις αλγόριθμοι για την επεξεργασία των TD: Ένας γενικός αλγόριθμος, ένας αλγόριθμος για non-isochronous TD, και ένας αλγόριθμος για isochronous TD.

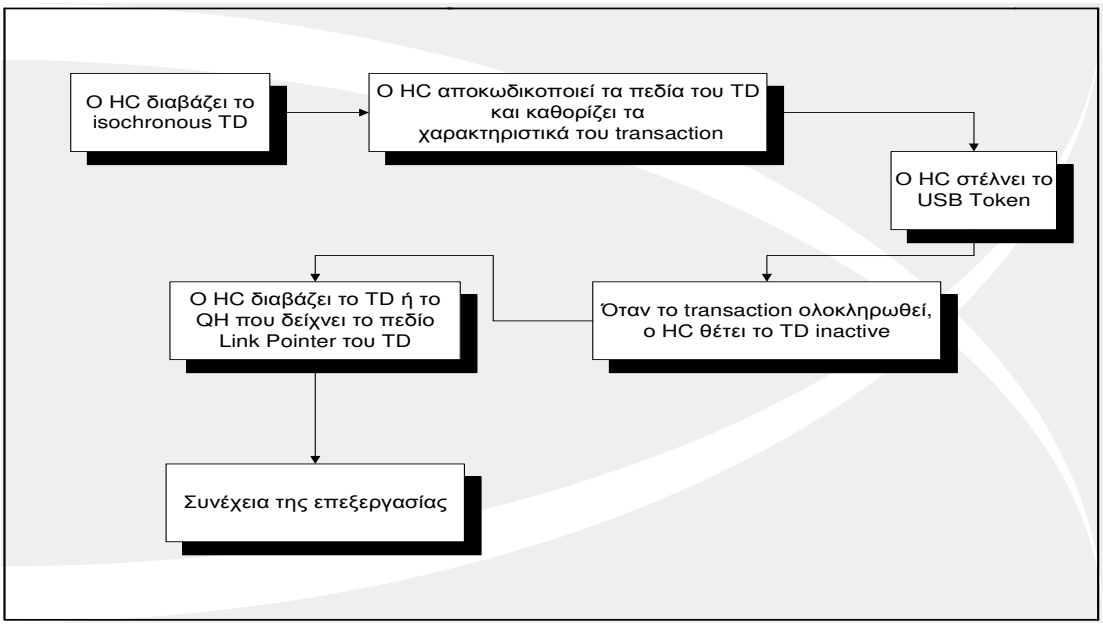


Σχήμα 4.4.2: Γενικός Αλγόριθμος επεξεργασίας των TD



Σχήμα 4.4.3: Αλγόριθμος επεξεργασίας των Interrupt, Control και Bulk TD

Σημειώνεται ότι στους αλγόριθμους των Σχημάτων 4.4.3 και 4.4.4 υπονοούνται τα βήματα του γενικού αλγόριθμου επεξεργασίας για την πραγματοποίηση των transactions και την ενημέρωση του status των TD.



Σχήμα 4.4.4: Αλγόριθμος επεξεργασίας των Isochronous TD

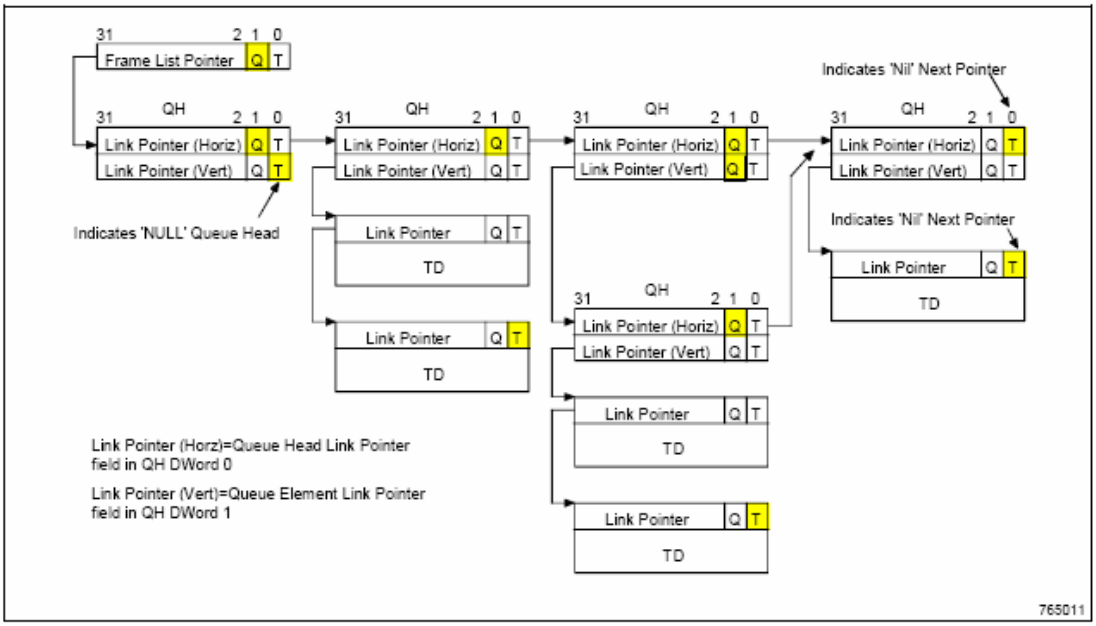
4.4.2 Επεξεργασία των Ουρών του Schedule

Με τον όρο Transfer Queuing ονομάζεται η δυνατότητα του HC να χειρίζεται δομές δεδομένων (TD και QH) οι οποίες είναι τοποθετημένες σε ουρές. Οι ουρές αυτές υλοποιούνται για την πραγματοποίηση εγγυημένης μεταφοράς δεδομένων. Έτσι όλα τα TD που αντιστοιχούν σε Interrupt, Control και Bulk transfers πρέπει να τοποθετούνται σε ουρές.

Στην κορυφή μιας ουράς βρίσκεται πάντα ένα Queue Head. Το QH αποτελείται από δύο link pointers. Το πρώτο link pointer είναι οριζόντιο (Queue Head Link Pointer) και συνδέει την ουρά του QH με την επόμενη ουρά του Schedule ή ένα TD. Αν το T bit του link pointer είναι 1, τότε το QH είναι η τελευταία δομή του frame. Το δεύτερο link pointer είναι κατακόρυφο (Queue Element Link Pointer) και δείχνει την πρώτη δομή δεδομένων (QH ή TD) που περιέχεται στην ουρά. Αν το T bit του link pointer είναι 1, τότε η ουρά είναι άδεια.

Στο Σχήμα 4.4.5 δείχνει τέσσερα παραδείγματα ουρών. Το πρώτο παράδειγμα δείχνει μια άδεια ουρά. Ο κατακόρυφος δείκτης έχει το T bit 1, ενώ ο οριζόντιος δείχνει στο επόμενο QH του Schedule. Το δεύτερο παράδειγμα δείχνει την αναμενόμενη τυπική μορφή μιας ουράς. Ο κατακόρυφος δείκτης δείχνει το πρώτο TD μιας συνδεδεμένης λίστας από TD. Ο οριζόντιος δείκτης δείχνει το QH της επόμενης ουράς.

Στο τρίτο παράδειγμα φαίνεται ότι, αν και συνήθως δεν είναι χρήσιμο, ο κατακόρυφος δείκτης ενός QH μπορεί να δείχνει σε ένα άλλο QH. Έτσι, όταν ο HC συναντήσει το ανώτερο QH, θα ακολουθήσει τον κατακόρυφο δείκτη και θα μπεί στο Q Context του κατώτερου QH. Το τελευταίο παράδειγμα δείχνει τη μορφή του τελευταίου QH ενός frame, που έχει το T bit του οριζόντιου δείκτη ίσο με 1.



Σχήμα 4.4.5 Τυπικά Παραδείγματα Ουρών

Σε κάθε ουρά, κάθε στοιχείο της ουράς δείχνει στο επόμενο στοιχείο μέσα σε αυτή. Ο HC ακολουθεί τον κατακόρυφο δείκτη του QH για να βρει το πρώτο TD της ουράς και το εκτελεί. Αν μετά την εκτέλεση το status του TD ικανοποιεί κάποια κριτήρια, τα οποία ονομάζονται "Advance Criteria", ο HC προχωράει *(advances)* μέσα στην ουρά γράφοντας τον link pointer του TD που μόλις εκτελέστηκε στο πεδίο του κατακόρυφου δείκτη (Queue Element Link Pointer) του QH της ουράς. Την επόμενη φορά που ο HC διασχίζει την ουρά, το επόμενο TD από αυτό που εκτελέστηκε θα είναι το TD που θα βρίσκεται στην κορυφή της ουράς των TD. Τα Advance Criteria παρουσιάζονται στον Πίνακα 4.4.1. Όταν το status ενός TD δείχνει σφάλματα στο transaction, τα Advance Criteria δεν ικανοποιούνται, ακομά και αν το όριο λαθών του TD δεν ξεπεράστηκε. Με τον τρόπο αυτό γίνεται έλεγχος λαθών και τα δεδομένα που μεταφέρονται είναι εγγυημένα σωστά.

Πίνακας 4.4.1: Queue Advance Criteria

Function-to-Host (IN) Transaction		**Host-to-Function (OUT) Transaction**	
Error/NAK	**Else**	**Error/NAK**	**Else**
Retry	Advance	Retry	Advance

Οπως έχει ήδη αναφερθεί, η διάσχιση των ουρών γίνεται με δύο μεθόδους: Breadth-First (default) και Depth First. Η μέθοδος διάσχισης, καθορίζεται από το Vf bit του Link Pointer του κάθε TD. Όταν η εκτέλεση ενός TD δεν ικανοποιεί τα Advance Criteria, το επόμενο Data Object βρίσκεται με διασχιση Breadth First. Αυτό σημαίνει ότι όταν ένα TD αποτυγχάνει, η ουρά του δεν γίνεται advance και ο HC διαβάζει το Queue Head Link Pointer του QH για να βρει το επόμενο TD για εκτέλεση. Ανεξάρτητα από τη μέθοδο διάσχισης, όταν ένα TD που βρίσκεται σε μια

ουρά εκτελείται επιτυχώς, ενημερώνει το QH της ουράς του, γράφοντας τον link pointer του στο πεδίο Queue Element Link Pointer του QH.

Στον πίνακα 4.4.2 φαίνονται οι πιθανοί συνδυασμοί των bits των link pointers και των κινήσεων για την εκτέλεση του Schedule, όταν τα Advance Criteria ικανοποιούνται.

Πίνακας 4.4.2: Πίνακας Απόφασης Διάσχισης του Schedule

#	Q Context	QH Q	QH T	QE Q	QE T	TD Vf	TD Q	TD T	Description
1	0	-	-	-	-	x	0	0	Όχι σε ουρά Εκτέλεση του TD TDLP για επόμενο TD
2	0	-	-	-	-	x	x	1	Όχι σε ουρά Εκτέλεση του TD Τέλος του frame
3	0	-	-	-	-	x	1	0	Όχι σε ουρά Εκτέλεση του TD TDLP για επόμενο (QH+QE) Qcontext=1*
4	1	0	0	0	0	0	x	x	Σε ουρά QELP για TD Εκτέλεση του TD Update του QELP με TDLP QHLP για επόμενο TD
5	1	x	x	0	0	1	0	0	Σε ουρά QELP για TD Εκτέλεση του TD Update του QELP με TDLP TDLP για επόμενο TD
6	1	x	x	0	0	1	1	0	Σε ουρά QELP για TD Εκτέλεση του TD Update του QELP με TDLP TDLP για επόμενο (QH+QE)
7	1	0	0	x	1	x	x	x	Σε ουρά Άδεια ουρά QHLP για επόμενο TD
8	1	x	x	1	0	-	-	-	Σε ουρά QELP για (QH+QE)
9	1	x	1	0	0	0	x	x	Σε ουρά QELP για TD Εκτέλεση του TD Update του QELP με TDLP Τέλος του frame
10	1	x	1	x	1	x	x	x	Σε ουρά Άδεια ουρά

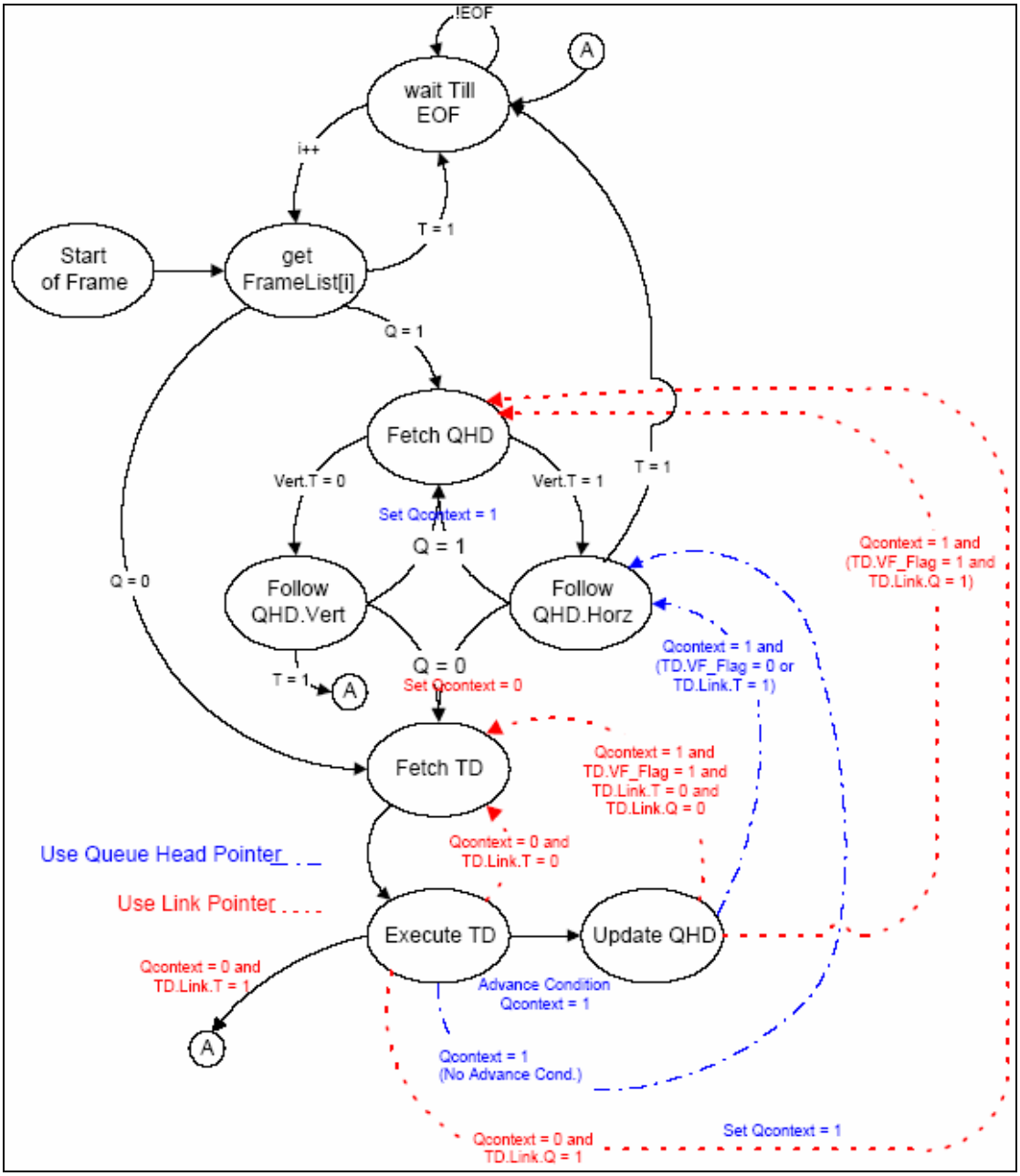
									Τέλος του frame
11	1	1	0	0	0	0	x	x	Σε ουρά QELP για TD Εκτέλεση του TD Update του QELP με TDLP QHLP για επόμενο (QH+QE)
12	1	1	0	x	1	x	x	x	Σε ουρά Άδεια ουρά QHLP για επόμενο (QH+QE)

*Ο HC θέτει κατάλληλα έναν εσωτερικό του καταχωρητή, ανάλογα με το αν η δομή δεδομένων που επεξεργάζεται βρίσκεται μέσα σε ένα Q Context ή εκτός Q Context.

Παρακάτω εξηγείται ο συμβολισμός που χρησιμοποιείται στον Πίνακα 4.4.2:

QHLP = Queue Head Link Pointer
QELP = Queue Element Link Pointer
TDLP = TD Link Pointer
QH.Q = Q bit του QHLP
QH.T = T bit του QHLP
QE.Q = Q bit του QELP
QE.T = T bit του QELP
TD.Vf = Vf bit του TDLP
TD.Q = Q bit του TDLP
X = αδιάφορο

Ένας εναλλακτικός τρόπος παρουσίασης του τρόπου διάσχισης του Schedule είναι το παρακάτω διάγραμμα. Το διάγραμμα αυτό είναι πιο πλήρες από τον αντίστοιχο πίνακα, καθώς περιέχει πρόσβαση στις δομές από οποιοδήποτε δείκτη (πχ από δείκτη του Frame List).



Σχήμα 4.4.6: Διάγραμμα Διάσχισης του Schedule

4.5 Διακοπές (Interrupts)

Ο Host Controller του UHCI παρέχει τη δυνατότητα παραγωγής διακοπών (*Interrupts*) από διάφορες πηγές (*Interrupt Sources)*. Υπάρχουν δύο ομάδες από Interrupt Sources: Αυτές που είναι αποτέλεσμα της εκτέλεσης των transactions του Schedule *(Transaction Based Interrupts)* και εκείνες που είναι αποτέλεσμα λανθασμένης λειτουργίας του HC *(Host Controller Operational Error)* και δεν έχουν σχέση με τα transactions του Schedule *(Non-Transaction Based Interrupts)*.

Όλες οι Transaction Based διακοπές μπορούν να γίνουν enabled/disabled *(maskable)* μέσω του USB Interrupt Enable Register. Οι διακοπές αυτές αναβάλλονται μέχρι την ολοκλήρωση του Frame. Μερικές φορές οι διακοπές στέλνονται μετά την εκπομπή του SOF πακέτου του επόμενου Frame, εγγυώντας με τον τρόπο αυτό ότι οι διακοπές επεξεργάζονται με ασφάλεια. Αντίθετα, εάν οποιαδήποτε στιγμή προκύψει HC Process Error ή Host System Error (Fatal Errors), ο HC σταματά *(halts)* αμέσως και στέλνει μια διακοπή υλικού στο σύστημα.

Η διαδικασία διαχείρισης των διακοπών *(Interrupt Handling)* είναι ίδια για όλες τις διακοπές. Όταν συμβεί μια διακοπή, ο έλεγχος πηγαίνει στον Interrupt Handler του HCD, ο οποίος διαβάζει το status από τον HC, συμπεραίνει το είδος της διακοπής, και εξυπηρετεί κατάλληλα τη διακοπή.

4.5.1 Διακοπές Προκαλούμενες από Συναλλαγές

Ο παρακάτω πίνακας συνοψίζει τις διακοπές που προκαλούνται από την εκτέλεση των transactions του Schedule και τις συνέπειες στο TD που αντιστοιχεί στο transaction και στους καταχωρητές του HC. Λεπτομέρειες για τις αιτίες των σφαλμάτων δίνονται στο Κεφάλαιο 2 και το Παράρτημα Α.

Πίνακας 4.5.1: Transaction Based Interrupts

Interrupt	Result
CRC Error / Time-out	Η τιμή του πεδίου C_Err του TD μειώνεται κατά 1. Όταν το C_Err γίνει 0 θέτονται: Active bit=0, Stalled bit=1, CRC/Timeout bit=1. Στο τέλος του frame το USB Error Interrupt bit του USBSTS τίθεται 1. Αν η διακοπή είναι enabled στον USBINTR θα σταλεί διακοπή υλικού.
Interrupt On Completion (IOC)	Το IOC bit του TD πρέπει να είναι 1. Στο τέλος του frame το USB Interrupt bit του USBSTS τίθεται 1. Αν στο τέλος του frame το active bit του TD είναι 0 τότε το IOC bit του USBSTS τίθεται 1. Αν η διακοπή είναι enabled στον USBINTR θα σταλεί διακοπή υλικού. Επιπλέον, αν το TD ολοκληρωθεί με σφάλματα το USB Error Interrupt bit του USBSTS θα τεθεί 1.
Short Packet Detect (SPD)	Το SPD bit του TD πρέπει να είναι 1. Στο τέλος του frame το USB Interrupt bit του USBSTS τίθεται 1. Αν η διακοπή

	είναι enabled στον USBINTR θα σταλεί διακοπή υλικού.
Serial Bus Babble	Active bit=0, Stalled bit=1, Babble bit=1. Το πεδίο C_Err δεν μειώνεται. Στο τέλος του frame το USB Error Interrupt bit του USBSTS τίθεται 1. Στέλνεται διακοπή υλικού. Εάν ένα EOF babble προκλήθηκε από τον HC, ο HC θα προκαλέσει ένα Bit Stuff Error ακολουθούμενο από ένα EOP στην αρχή του απόμενου frame.
Stalled	Stall bit=1, Active bit=0. Το πεδίο C_Err δεν μειώνεται. Στέλνεται διακοπή υλικού. Το HCD δεν πρέπει να δέχεται αιτήσεις για το endpoint που είναι Stalled, μέχρι να αλλάξει η κατάστασή του.
Data Buffer Error	Η τιμή του πεδίου C_Err του TD μειώνεται κατά 1. Όταν το C_Err γίνει 0 θέτονται: Active bit=0, Stalled bit=1. Στο τέλος του frame το USB Error Interrupt bit του USBSTS τίθεται 1. Στέλνεται διακοπή υλικού.
Bit Stuff Error	Η τιμή του πεδίου C_Err του TD μειώνεται κατά 1. Όταν το C_Err γίνει 0 θέτονται: Active bit=0, Stalled bit=1. Στο τέλος του frame το USB Error Interrupt bit του USBSTS τίθεται 1. Στέλνεται διακοπή υλικού.

4.5.2 Άλλες Διακοπές

Οι διακοπές της κατηγορίας αυτής είναι: Resume Received, Host Controller Process Error και Host System Error. Παρακάτω παρουσιάζονται οι καταστάσεις που εγείρουν τις διακοπές αυτές και οι συνέπειες των διακοπών.

4.5.2.1 Διακοπή Resume Received

Η διακοπή Resume Received στέλνεται όταν ο HC λαμβάνει ένα σήμα RESUME από μια συσκευή στο USB όταν βρίσκεται σε κατάσταση Global Suspend. Εάν η διακοπή είναι enabled στον USBINTR, στέλνεται στο σύστημα διακοπή υλικού, επιτρέποντας στο σύστημα να βγεί από το Suspend State και να γυρίσει στην κανονική λειτουργία.

4.5.2.2 Διακοπή Host Controller Process Error

Ο HC ελέγχει ορισμένα κρίσιμα πεδία των TD κατά τη λειτουργία του για να εξασφαλίσει ότι τα δεδομένα που χειρίζεται είναι ορθά. Τα πεδία αυτά είναι τα PID και MaxLength, στα οποία επιτρέπονται μόνο συγκεκριμένες τιμές. Εάν οι τιμές των πεδίων αυτών είναι μη αποδεκτές, ο HC σταματά αμέσως τη λειτουργία του, θέτει 1 το HC Process Error bit του USBSTS και στέλνει μια διακοπή υλικού στο σύστημα. Εάν το transaction που αντιστοιχεί στο μη ορθό TD έχει ήδη ξεκινήσει στο USB, ο HC πρέπει να προκαλέσει μια κατάσταση σφάλματος (πχ bit stuff error) και μετά να

σταματήσει. Η διακοπή αυτή δεν μπορεί να απενεργοποιηθεί από τον USBINTR. Το HCD ευθύνεται για την επαναφορά του HC σε κανονική λειτουργία.

4.5.2.3 Διακοπή Host System Error

Σε ένα PCI σύστημα, ο HC θέτει 1 το Host System Error bit του USBSTS όταν συμβεί ένα από τα εξής σοβαρά σφάλματα: PCI Parity Error, PCI Master Abort, PCI Target Abort. Η περιγραφή των σφαλμάτων αυτών είναι εκτός του σκοπού του παρόντος συγγράμματος. Όταν συμβεί τέτοιο σφάλμα, ο HC θέτει 0 το Run/Stop bit του USBCMD για να σταματήσει την εκτέλεση του Schedule. Η διακοπή αυτή δεν μπορεί να απενεργοποιηθεί από τον USBINTR. Η αντιμετώπιση αυτού του σφάλματος είναι ευθύνη του Λειτουργικού Συστήματος και όχι του HCD.

ΚΕΦΑΛΑΙΟ 5

Λογισμικό Οδήγησης του UHCI

Στο Κεφάλαιο αυτό περιγράφεται λεπτομερώς το λογισμικό που αναπτύχθηκε στα πλαίσια της Διπλωματικής Εργασίας για την οδήγηση ενός Host Controller συμβατού με το πρότυπο UHCI. Το λογισμικό αυτό δεν αποτελεί ένα πλήρες Host Controller Driver. Όμως υλοποιεί τις πιο σημαντικές συνιστώσες ενός UHCI Host Controller Driver. Το παρόν λογισμικό υλοποιεί τα παρακάτω:

- Όλες τις απαραίτητες δομές δεδομένων για την υλοποίηση των αιτήσεων μεταφοράς δεδομένων του Συστήματος USB (Pipes, IRPs, URBs).

- Όλες τις απαραίτητες δομές δεδομένων για την οδήγηση του UHCI Host Controller (TDs, QHs, Frame List).

- Αλγόριθμοι διαχείρισης των αιτήσεων μεταφοράς δεδομένων με απόδοση δικαιοσύνης σε όλα τα Pipes.

- Αλγόριθμοι χρονοδρομολόγησης των αιτήσεων μεταφοράς δεδομένων και κατασκευής του Schedule του Host Controller.

- Αλγόριθμοι ανάγνωσης του Status κάθε transactions και επιστροφής του στα αντίστοιχα IRP.

- Συναρτήσεις ανάγνωσης και εγγραφής όλων των καταχωρητών του UHCI.

- Συναρτήσεις εύρεσης των UHCI Host Controllers στον δίαυλο PCI ενός υπολογιστικού συστήματος.

Παρ' όλα αυτά η (έστω πειραματική) οδήγηση ενός πραγματικού Host Controller και ο έλεγχος της ορθότητας του λογισμικού δεν κατέστησαν δυνατά για τους παρακάτω λόγους:

- Δεν αναπτύχθηκαν συναρτήσεις διαχείρισης των διακοπών υλικού του Host Controller. Έτσι, το λογισμικό δεν μπορεί μπορεί να αναγνωρίσει και να διαχειριστεί τα γεγονότα που αναφέρονται από το υλικό αποκλειστικά μέσω διακοπών.

- Δεν αναπτύχθηκε λογισμικό οδήγησης του Root Hub. Έτσι, δεν είναι δυνατή η ροή δεδομένων μεταξύ του Host Controller και των συσκευών που είναι συνδεδεμένες στο USB.

- Η πραγματική μεταφορά δεδομένων απαιτεί την ύπαρξη ενός USBD (*Universal Serial Bus Driver*) που να διαχειρίζεται τους buffers όπου αποθηκεύονται τα δεδομένα προς αποστολή/λήψη. Έτσι το υλικό δεν λαμβάνει δεδομένα στους buffers και εγείρει μια διακοπή Host System Error σταματώντας την λειτουργία του, καθιστώντας αδύνατη την οδήγησή του.

- Απαιτούνται Client SWs που να οδηγούν τις συσκευές που συνδέονται στο USB και να παράγουν αιτήσεις μεταφοράς δεδομένων ανάλογα με τις ανάγκες κάθε συσκευής.

Η υλοποίηση των παραπάνω συνιστωσών του συστήματος USB θα έχουν ως αποτέλεσμα την επιτυχή οδήγηση του Host Controller. Ο έλεγχος της ορθότητας του

λογισμικού πραγματοποιήθηκε με την ανάπτυξη λογισμικού που προσομοιώνει την λειτουργία ενός Host Controller, επιστρέφοντας ένα εικονικό status για κάθε transaction που χρονοδρομολογείται στον δίαυλο. Στο υπόλοιπο του παρόντος κεφαλαίου περιγράφεται λεπτομερώς το λογισμικό που αναπτύχθηκε για την υλοποίηση του HCD και τον έλεγχο της ορθότητάς του. Ο πηγαίος κώδικας του λογισμικού περιέχεται στο συνοδευτικό CD-ROM της Διπλωματικής Εργασίας (βλέπε Παράρτημα Δ).

Στις ενότητες που ακολουθούν περιγράφεται η υλοποίηση των διαφόρων δομών δεδομένων που χρησιμοποιεί το HCD, καθώς και οι αλγόριθμοι που υλοποιήθηκαν για την εκτέλεση των διαφόρων επεξεργασιών που είναι απαραίτητες για την οδήγηση του Host Controller και τον έλεγχο της ορθότητας των αλγορίθμων που αναπτύχθηκαν.

5.1. Υλοποίηση των Αιτήσεων Μεταφοράς Δεδομένων

Στις επόμενες παραγράφους παρουσιάζεται η υλοποίηση των Pipes, IRPs και URBs του συστήματος USB. Θεωρείται ότι το USBD παρέχει τα IRP προς εκτέλεση στο HCD. Το HCD αναλαμβάνει να εξασφαλίσει την «δικαιοσύνη» στα Pipes, δηλαδή να εκτελέσει όλα τα IRPs που εκκρεμούν από όλα τα Pipes, με σειρά που να μην ευνοεί κάποιο Pipe. Το HCD αντιστοιχεί κάθε IRP στα URBs του και στη συνέχεια τα τοποθετεί σε κατάλληλες ουρές για την χρονοδρομολόγησή τους.

5.1.1 Υλοποίηση των Σωληνώσεων (Pipes)

Τα Pipes υλοποιούνται ως δομές δεδομένων. Κάθε Pipe περιέχει όλες τις απαραίτητες πληροφορίες για την επικοινωνία με το Endpoint που αντιπροσωπεύει το Pipe:

- Τη διεύθυνση της συσκευής και τον αριθμό του Endpoint που σχετίζεται με το Pipe.
- Τον τύπο μεταφορές δεδομένων (Isochronous, Interrupt, Control ή Bulk), την ταχύτητα (low-speed ή full-speed), και την κατεύθυνση που υποστηρίζει το Endpoint.
- Την Περίοδο Εξυπηρέτησης *(Polling Interval)* του Endpoint, αν αυτό είναι Interrupt. Η περίοδος εξυπηρέτησης δείχνει την συχνότητα κατά την οποία το Endpoint διαθέτει δεδομένα για μεταφορά προς τον Host ή είναι έτοιμο να δεχθεί δεδομένα από τον Host.
- Το μέγιστο Data Payload που υποστηρίζει το Endpoint.
- Την τρέχουσα τιμή του Sequence Bit του Pipe, αν αυτό δεν είναι Isochronous.
- Τον αριθμό του Host Controller με τον οποίο επικοινωνεί το Endpoint. Σε ένα σύστημα μπορούν να υπάρχουν περισσότεροι από ένας HCs. Για τον λόγο

αυτό όταν το λογισμικό ελέγχει την παρουσία υλικού, αναθέτει σε κάθε Host Controller έναν αριθμό.

- Την ταυτότητα του Pipe (*Pipe ID*), ένας μοναδικός ακέραιος για κάθε Pipe.

Τα συνολικά Pipes που υποστηρίζει ένα USB σύστημα είναι MAX_DEVS· 4096, όπου MAX_DEVS είναι ο μέγιστος αριθμός HCs που μπορεί να υποστηρίξει το σύστημα. Θεωρητικά δεν υπάρχει κάποιο όριο για το MAX_DEVS. Ο παράγοντας 4096 προκύπτει από τον μέγιστο αριθμό Endpoints σε κάθε δίαυλο USB (16 Endpoints· 128 Συσκευές· 2 για κάθε κατεύθυνση).

5.1.2 Υλοποίηση των IRPs

Τα IRPs υλοποιούνται ως δομές δεδομένων. Κάθε IRP περιέχει όλες τις απαραίτητες πληροφορίες που σχετίζονται με μια αίτηση μεταφοράς δεδομένων:

- Την ταυτότητα του Pipe στο οποίο αναφέρεται το IRP.
- Την διεύθυνση της αρχής του buffer στο οποίο είναι αποθηκευμένα τα δεδομένα προς αποστολή, ή θα αποθηκευθούν τα δεδομένα που θα ληφθούν.
- Το μέγεθος των δεδομένων που θα αποσταλούν/ληφθούν.
- Την ταυτότητα του IRP (*IRP ID*), ένας μοναδικός ακέραιος για κάθε IRP.
- Το επιθυμητό frame εκτέλεσης, αν το IRP ανήκει σε Isochronous Pipe.
- Το όριο επαναλήψεων της εκτέλεσης κάθε URB του IRP σε περίπτωση σφάλματος.

Επίσης, για κάθε IRP δεσμέυεται μνήμη για την αποθήκευση του τελικού status της εκτέλεσής του, και τον αριθμό των URB στα οποία διασπάται.

Το USBD τοποθετεί τα IRP που λαμβάνει από τα Clients σε συνδεδεμένες λίστες. Κάθε Pipe έχει τη δική του συνδεδεμένη λίστα από IRP, στην οποία τοποθετούνται IRP μόνο του συγκεκριμένου Pipe. Το HCD μπορεί να διασχίσει την λίστα κάθε Pipe ώστε να εκτελέσει τα IRP.

5.1.3 Υλοποίηση των URBs

Τα URBs υλοποιούνται ως δομές δεδομένων. Κάθε URB περιέχει όλες τις απαραίτητες πληροφορίες για τη δημιουργία του αντίστοιχου Trasfer Descriptor, έτσι ώστε να τοποθετηθεί στο Schedule και να εκτελεστεί από τον HC. Έτσι κάθε URB περιέχει τα παρακάτω:

- Όλες τις πληροφορίες του Pipe στο οποίο ανήκει.
- Το PID του TD που θα προκύψει από το URB.

- Το όριο επαναλήψεων της εκτέλεσης του TD.
- Το polling interval, αν είναι Interrupt URB.
- Την διεύθυνση της αρχής του buffer στο οποίο είναι αποθηκευμένα τα δεδομένα προς αποστολή, ή θα αποθηκευθούν τα δεδομένα που θα ληφθούν.
- Το μέγεθος των δεδομένων που θα αποσταλούν/ληφθούν.
- Την ταυτότητα του IRP από το οποίο προέκυψε.
- Τον υπολογιζόμενο (αναμενόμενο) χρόνο διαύλου κατά την εκτέλεση του αντίστοιχου TD.
- Το επιθυμητό frame εκτέλεσής του, αν είναι Isochronous URB.

5.1.4 Λίστες URBs των Συσκευών

Τα URBs τοποθετούνται σε συνδεδεμένες λίστες για την επεξεργασία τους κατά την χρονοδρομολόγησή τους. Τα URBs τοποθετούνται στις λίστες σε σειρά FIFO. Έτσι, τα URBs εισάγονται στις λίστες με τη σειρά που παράγονται από τα IRPs, και με την ίδια σειρά χρονοδρομολογούνται στο Schedule. Τα Isochronous URBs τοποθετούνται στις λίστες ανάλογα με τον HC από τον οποίο εξυπηρετούνται. Δηλαδή το HCD διαθέτει MAX_DEVS λίστες για τα Isochronous URBs. Τα Interrupt URBs τοποθετούνται στις λίστες ανάλογα με τον HC από τον οποίο εξυπηρετούνται και την Διαίρεση του Schedule (*Schedule Division,* βλέπε παρακάτω) στην οποία ανήκουν. Το ίδιο ισχύει για τα Control και Bulk URBs.

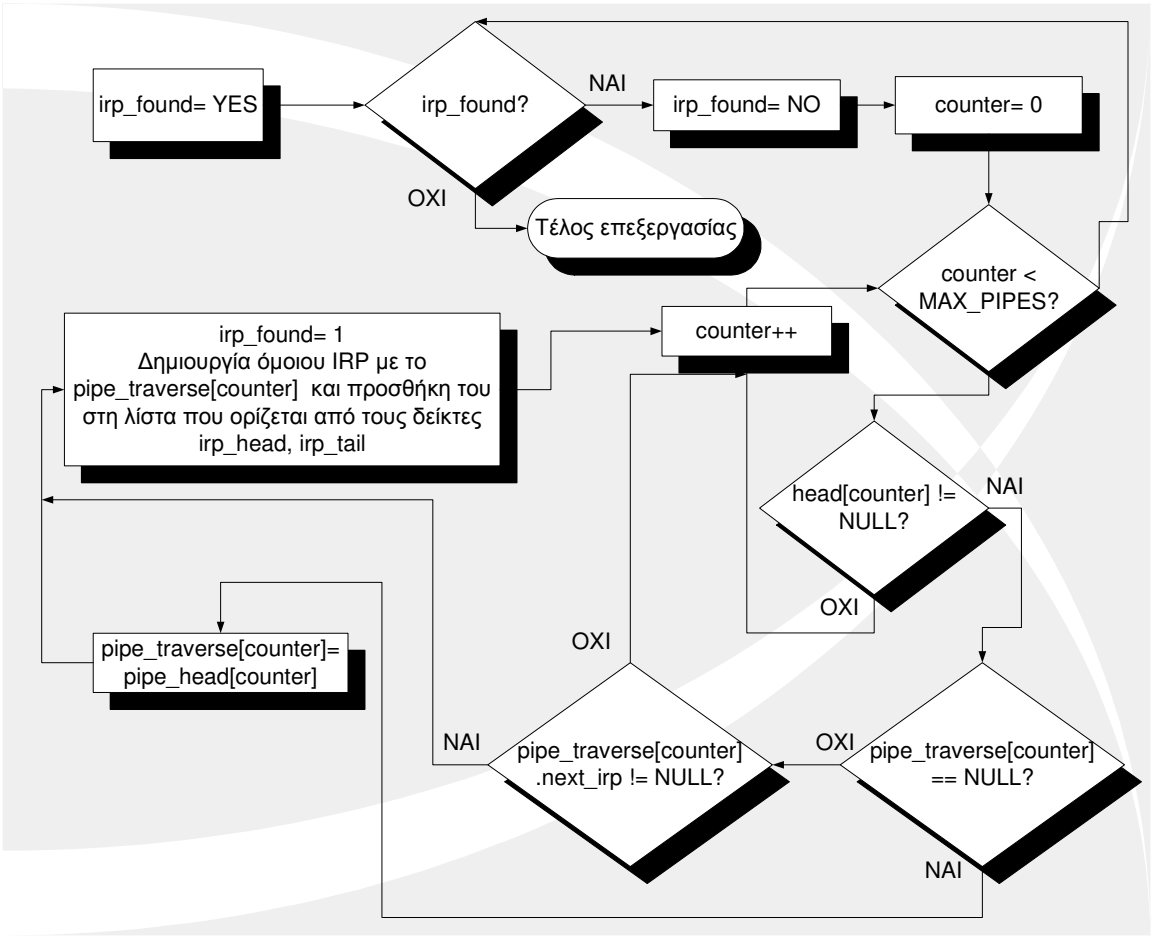
Το σύνολο των Pipes διαιρείται SCHED_DIV= 2 μέρη (*Schedule Divisions*), ανάλογα με κάποιο κριτήριο που να εξασφαλίζει την ανεξαρτησία των IRP του κάθε Schedule Division. Δηλαδή η εκτέλεση κάποιου IRP του ενός Division να μην επηρρεάζει την ορθότητα της εκτέλεσης οποιουδήποτε IRP του άλλου Division. Το κριτήριο που επιλέχθηκε στην παρούσα υλοποίηση είναι το αν το Pipe ID είναι άρτιο ή περιττό. Ο λόγος για τον οποίο είναι απαραίτητος αυτός ο διαχωρισμός των IRPs στην παρούσα υλοποίηση θα φανεί στην παράγραφο όπου παρουσιάζεται η υλοποίηση του Schedule.

5.1.5 Υλοποίηση της Δικαιοσύνης στα Pipes

Όπως έχει ήδη αναφερθεί το HCD πρέπει να εξυπηρετεί τα IRP όλων των Pipes με τέτοιο τρόπο έτσι ώστε να μην ευνοείται κάποιο pipe εις βάρος των άλλων. Αυτή η απαίτηση ικανοποιείται με τον αλγόριθμο που παρουσιάζεται στο Σχήμα 5.2.1.

Υποτίθεται ότι οι δείκτες head[i] και tail[i] δείχνουν το πρώτο και το τελευταίο στοιχείο της συνδεδεμένης λίστας των IRP του Pipe με ταυτότητα i. Ο δείκτης pipe_traverse[i] δείχνει το τελευταίο στοιχείο της λίστας του Pipe i που έχει διαβαστεί (ή NULL αν η λίστα είναι κενή ή δεν έχει διαβαστεί ακόμα κάποιο στοιχείο). Ο

αλγόριθμος του σχήματος 5.1.1 διαβάζει κάθε φορά ένα στοιχείο (αν υπάρχει) από τη λίστα κάθε Pipe και το αντιγράφει σε μια νέα λίστα, που ορίζεται από τους δείκτες irp_head και irp_tail. Στη συνέχεια διαβάζει το επόμενο στοιχείο κάθε λίστας κοκ μέχρι να έχουν διαβαστεί όλα τα IRP που εκκρεμούν. Η νέα λίστα που προκύπτει θα περιέχει όλα τα IRP που εκκρεμούν στο σύστημα δίνοντας το ίδιο βάρος στα IRP όλων των Pipes. Δηλαδή ανάμεσα σε δύο διαδοχικά IRP του ίδιου Pipe θα υπάρχεί ένα IRP για κάθε ένα από όλα τα υπόλοιπα Pipes (εφόσον υπάρχουν IRPs για τα Pipe αυτά).



Σχήμα 5.1.1: Αλγόριθμος Διάσχισης των Λιστών IRP των Pipes

5.1.6 Αντιστοίχηση των IRPs σε URBs

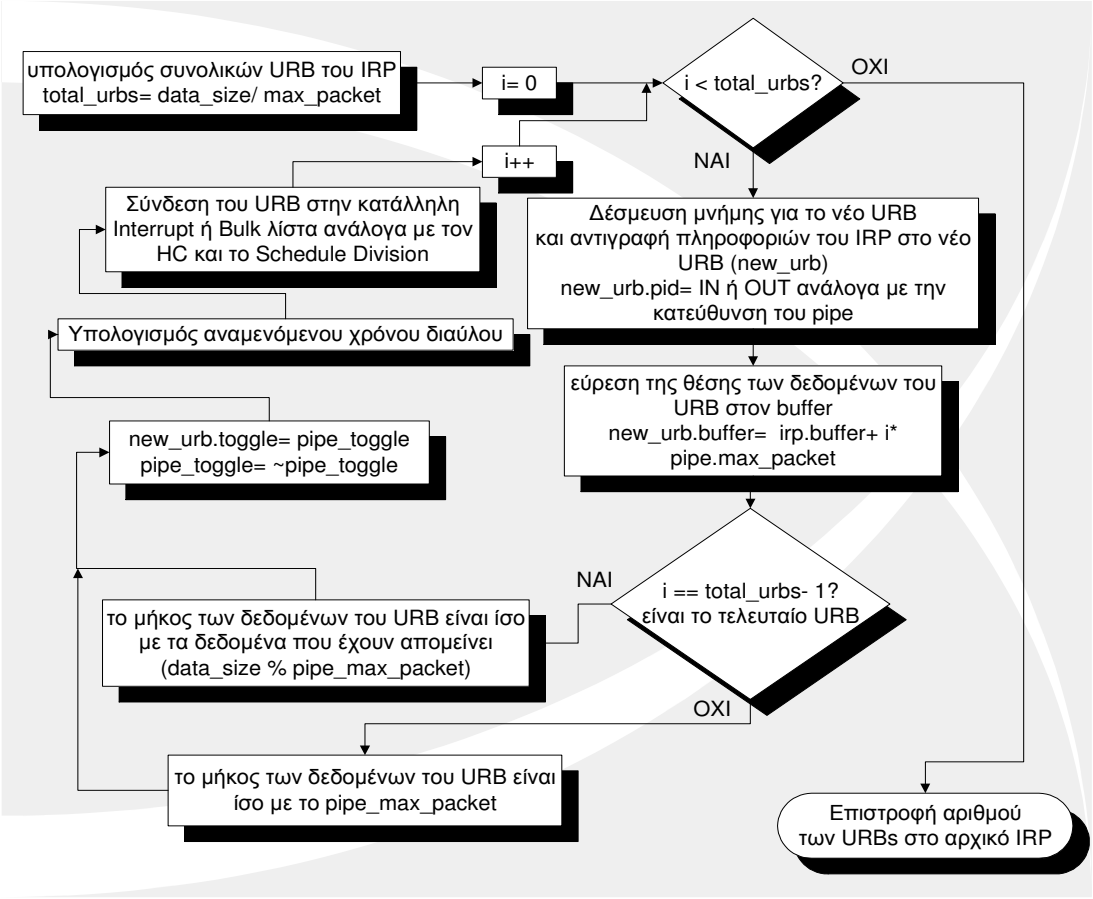
Όπως αναφέρθηκε στην προηγούμενη παράγραφο, το HCD διαβάζει τα εκκρεμή IRP από όλα τα Pipes και τα τοποθετεί στην συνδεδεμένη λίστα που ορίζεται από τους δείκτες irp_head και irp_tail. Στη συνέχεια, το HCD εξετάζει σειριακά τα IRP της λίστας αυτής και δημιουργεί για κάθε IRP τα αντίστοιχα URB, τα οποία τοποθετεί στην κατάλληλη URB λίστα, ανάλογα με τον τύπο μεταφοράς δεδομένων της αίτησης, τον HC που θα την εξυπηρετήσει, και το Schedule Division στο οποίο αναφέρεται (το τελευταίο κριτήριο δεν ισχύει για τα Isochronous IRP). Το HCD εξασφαλίζει ότι όλα τα URB ενός IRP θα τοποθετηθούν στην σωστή λίστα με τη

σειρά με την οποία δημιουργούνται. Στις επόμενες παραγράφους παρουσιάζονται οι αλγόριθμοι δημιουργίας των URBs ενός IRP ανάλογα με τον τύπο μεταφοράς δεδομένων που υποστηρίζει το Pipe στο οποίο ανήκει το υπό εξέταση IRP.

5.1.6.1 Interrupt και Bulk URBs

Ο αλγόριθμος που παρουσιάζεται στο σχήμα 5.1.2 πραγματοποιεί το «τεμάχισμα» ενός Interrupt ή Bulk IRP στα ισοδύναμα URB του. Ο αλγόριθμος αυτός βασίζεται στις παρατηρήσεις των Παραγράφων 2.3.4.1 και 2.3.4.3 όπου επεξηγείται η μορφή των Interrupt και Bulk transactions.

Αρχικά υπολογίζεται ο αριθμός των URBs που απαιτούνται για την ενθυλάκωση των δεδομένων που πρόκειται να μεταφερθούν. Όλα τα URBS που δημιουργούνται περιέχουν δεδομένα μεγέθους ίσου με το μέγιστο Data Payload που υποστηρίζεται από το Pipe εκτός από το τελευταίο URB που περιέχει το υπόλοιπο των δεδομένων ή δεδομένα μηδενικού μήκους (στην περίπτωση που το μήκος των δεδομένων είναι ακέραιο πολλαπλάσιο του μέγιστου Data Payload του Pipe). Η αρχή των δεδομένων του κάθε URB βρίσκεται στην θέση urb_no· max_packet του buffer του IRP, όπου urb_no είναι ο αριθμός του URB του IRP (ξεκινώντας την αρίθμιση από το 0) και max_packet είναι το μέγιστο Data Payload του Pipe. Στη συνέχεια το HCD εκτιμά την αναμενόμενη διάρκεια του Transaction του URB στον δίαυλο και τοποθετεί το URB στην κατάλληλη συνδεδεμένη λίστα για την χρονοδρομολόγησή του. Το HCD προσθέτει το κατάλληλο Sequence Bit σε κάθε URB έτσι ώστε να ικανοποιείται το πρωτόκολλο Toggle Sequencing. Τέλος το HCD επιστρέφει στο IRP που βρίσκεται στην λίστα των IRP του Pipe τον αριθμό των URBs στα οποία αντιστοιχεί.

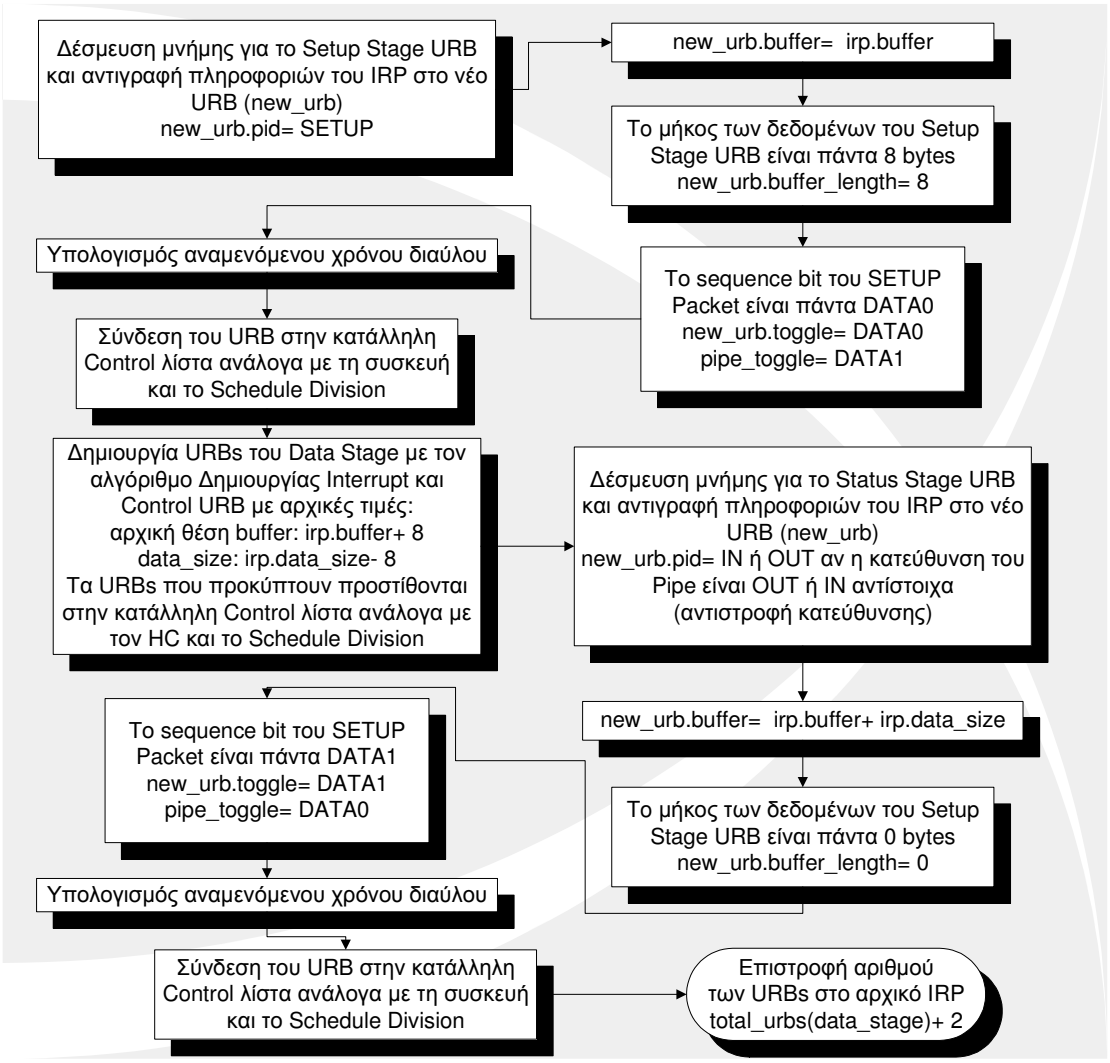


Σχήμα 5.1.2: Αλγόριθμος Δημιουργίας των Interrupt και Bulk URBs

5.1.6.2 Control URBs

Το «τεμάχισμα» ενός Control IRP στα ισοδύναμα URB του πραγματοποιείται μέσω του αλγορίθμου του σχήματος 5.1.3. Ο αλγόριθμος αυτός βασίζεται στις παρατηρήσεις της Παραγράφου 2.3.4.2 όπου επεξηγείται η μορφή των Control Transfers.

Αρχικά δημιουργείται το URB που αντιστοιχεί στο στάδιο Setup του control Transfer. Το μεγεθος των δεδομένων του URB αυτού είναι πάντα 8 bytes. Στη συνέχεια δημιουργούνται τα URB του σταδίου Data χρησιμοποιώντας έναν αλγόριθμο όμοιο με αυτόν του σχήματος 5.1.2, και τέλος δημιουργείται το URB του σταδίου Status, του οποίου το μήκος των δεδομένων είναι πάντα 0 bytes. Τα URBs που δημιουργούνται προστίθενται στην κατάλληλη Control λίστα των URBs. Για κάθε URB εκτιμάται ο χρόνος διαύλου του transaction που αντιπροσωπεύει, και ορίζεται το Sequence Bit του έτσι ώστε να μην παραβιάζονται οι κανόνες των Control Transfers. Τέλος το HCD επιστρέφει στο IRP που βρίσκεται στην λίστα των IRP του Pipe τον αριθμό των URBs στα οποία αντιστοιχεί.

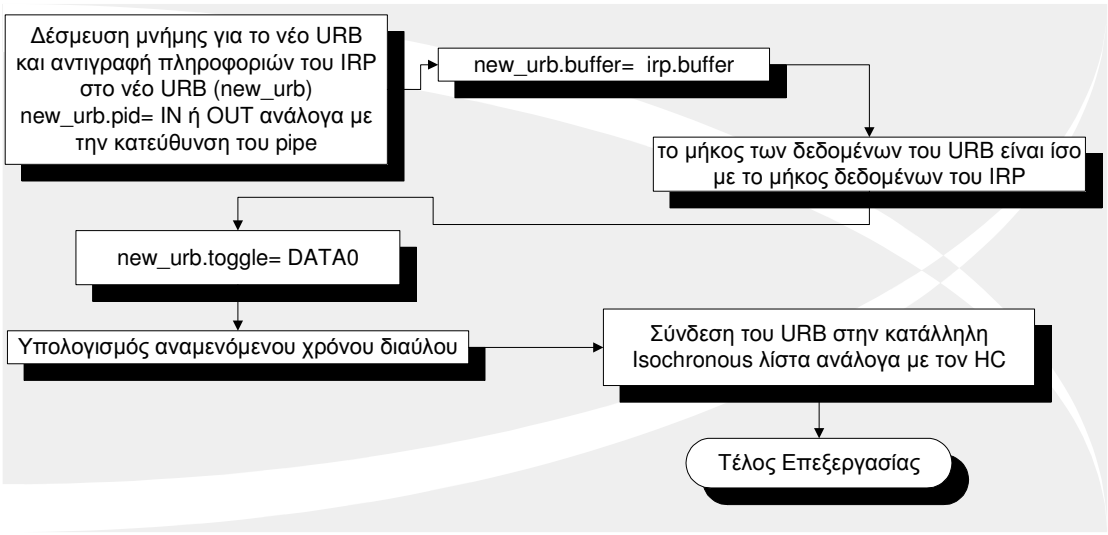


Σχήμα 5.1.3: Αλγόριθμος Δημιουργίας των Control URBs

5.1.6.3 Isochronous URBs

Ο αλγόριθμος του Σχήματος 5.1.4 παρουσιάζει τη δημιουργία ενός Isochronous URB από το αντίστοιχο IRP. Ένα Isochronous IRP ισοδυναμεί πάντα με ένα μόνο Isochronous URB. Ο αλγόριθμος που ακολουθεί βασίζεται στις παρατηρήσεις της Παραγράφου 2.3.4.4, όπου περιγράφεται η μορφή των Isochronous transactions.

Το HCD αντιγράφει στο νέο URB τις πληροφορίες του Isochronous IRP και υπολογίζει τον αναμενόμενο χρόνο διαύλου του transaction που αντιπροσωπεύει το URB. Στη συνέχεια συνδέει το URB στην λίστα των Isochronous URBs του HC που θα πραγματοποιήσει τη μεταφορά. Το Sequence Bit όλων των Isochronous URBs τίθεται ίσο με DATA0, όπως απαιτείται. Στις ισόχρονες μεταφορές δεν υποστηρίζεται συγχρονισμός Toggle Sequencing.



Σχήμα 5.1.4: Αλγόριθμος Δημιουργίας των Isochronous URBs

5.1.6.4 Υπολογισμούς του Χρόνου Διαύλου των URBs

‘Οπως περιγράφτηκε προηγουμένως, το HCD υπολογίζει για κάθε URB τον αναμενόμενο χρόνο διαύλου του transaction που εκπροσωπεί. Το HCD θα χρησιμοποιήσει αυτή την εκτίμηση της διάρκειας κάθε transaction για να χρονοδρομολογήσει κατάλληλα τις αιτήσεις μεταφοράς δεδομένων, χωρίς να παραβιάσει τον χρόνο του κάθε frame και τον διαθέσιμο χρόνο για κάθε είδος μεταφοράς δεδομένων.

Ο υπολογισμός του αναμενόμενου χρόνου των transactions υλοποιείται χρησιμοποιώντας τις συναρτήσεις 3.2.1 έως 3.2.6 της Παραγράφου 3.2.2, ανάλογα με το είδος του transaction (Isochronous, Non Isochronous), την ταχύτητά του (Full-speed, Low-speed), και την κατεύθυνσή του (IN, OUT). Στις συναρτήσεις αυτές τίθονται απαισιόδοξες αλλά ρεαλιστικές τιμές για τις παραμέτρους τους. Έτσι τέθηκαν Host_Delay= 1 μs, Hub_LS_Setup= 400 ns (τιμή μεγαλύτερη από τον χρόνο τεσσάρων full-speed bits, όπως απαιτείται), και BitStuffTime= 1.1667· 8· Data_bc (χειρότερη περίπτωση), όπου Data_bc είναι το Data Payload του υπό εξέταση URB.

5.2 Υλοποίηση των Δομών Δεδομένων του UHCI

Στις επόμενες παραγράφους περιγράφεται ο τρόπος με τον οποίο υλοποιήθηκαν οι Δομές Δεδομένων του UHCI, και οι απαιτήσεις για τη δέσμευση μνήμης για τις δομές αυτές. Επίσης περιγράφεται ο τρόπος με τον οποίο το HCD διαχειρίζεται τις δομές αυτές για την κατασκευή του Schedule.

5.2.1 Υλοποίηση των Δομών Δεδομένων

Οι δομές δεδομένων υλοποιήθηκαν σύμφωνα με τις προδιαγραφές του UHCI, οι οποίες περιγράφονται στην Ενότητα 4.3. Έτσι το Frame List υλοποιήθηκε ως ένα array από 1024 Dwords, aligned στα όρια 4096 bytes, όπου κάθε Dword αντιπροσωπεύει το Frame List Pointer που αντιστοιχεί στο συγκεκριμένο entry του Frame List. Κάθε TD υλοποιείται ως μια δομή με τα τέσσερα απαιτούμενα Dwords: TD Link Pointer, TD Control & Status, TD Token, TD Buffer Pointer. Στα υπόλοιπα τέσερα Dwords προστέθηκαν πεδία για πληροφορίες όπως την ταυτότητα του IRP στο οποίο ανήκει, τον αριθμό του URB στο οποίο αντιστοιχεί, το pipe στο οποίο ανήκει κτλ. Η τελική δομή που προκύπτει ικανοποιεί την απαίτηση να είναι στοιχισμένη (*Aligned*) στα όρια 16 bytes. Τέλος κάθε Queue Head υλοποιείται ως μια δομή με τα δύο απαιτούμενα Dwords, Queue Head Link Pointer και Queue Element Link Pointer, η οποία είναι aligned στα όρια 16 bytes.

5.2.2 Δέσμευση Μνήμης για τις Δομές Δεδομένων

Το λογισμικό χρησιμοποιεί τις παραπάνω δομές δεδομένων για την κατασκευή του χρονοδιαγράμματος (*Schedule*). Έτσι δεσμεύει την απαραίτητη μνήμη για τη χρησιμοποίηση των παραπάνω δομών. Συγκεκριμένα το HCD χρησιμοποιεί:

- Ένα Frame List για κάθε Host Controller.

- 150 Transfer Descriptors για κάθε Frame και για κάθε Host Controller. Ο μέγιστος αριθμός transactions που μπορούν να πραγματοποιηθούν σε ένα frame είναι 150, όπως φαίνεται από τους πίνακες 2.2.1 έως 2.2.6 των παραγράφων 2.2.5 έως 2.2.8. Έτσι σε κάθε frame απαιτούνται το πολύ 150 TDs για την περιγραφή των εκκρεμών transactions.

- 15 Queue Heads για κάθε Schedule Division και για κάθε Host Controller. Δηλαδή συνολικά 30 QHs για κάθε Host Controller. Ο λόγος για τον οποίο απαιτείται αυτός ο αριθμός από QHs θα φανεί στην επόμενη παράγραφο.

Το HCD αποθηκεύει τις διευθύνσεις των Transfer Descriptors σε ένα array για κάθε Host Controller και για κάθε Frame, έτσι ώστε να μπορεί να τις προσπελάσει ανά πάσα στιγμή. Επίσης διαθέτει έναν μετρητή για κάθε Frame που να μετράει τον αριθμό των TDs του Frame που έχουν ήδη χρησιμοποιηθεί. Με βάση την τιμή του μετρητή αυτού μπορεί να προσπελάσει το επόμενο TD προς επεξεργασία. Με τον ίδιο

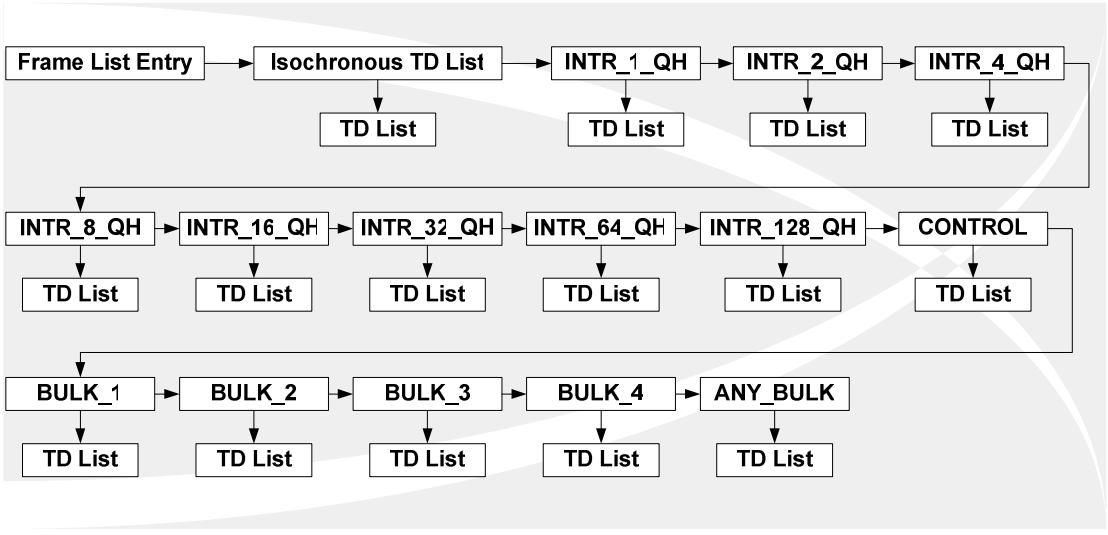
τρόπο αποθηκέυει τις διευθύνσεις των Queue Heads που έχει δεσμεύει για κάθε Host Controller και κάθε Schedule Division. Σε ό,τι αφορά το Frame List, το HCD αποθηκεύει μόνο την βασική διεύθυνση (*Frame List Base Address*). Έτσι μπορεί να προσπελάσει οποιοδήποτε entry (έστω το i-στο) του Frame List υπολογίζοντας την παράσταση (Frame List Base Address) + i · 4, αφού τα entries του Frame List είναι συνεχή και έχουν μέγεθος τεσσάρων bytes (Dwords).

Σημειώνεται ότι το HCD εξασφαλίζει ότι οι διευθύνσεις όλων των δομών δεδομένων θα έχουν τα τέσσερα τελευταία bits μηδενικά, όπως απαιτείται από το πρότυπο.

5.2.3 Σκελετός των Queue Heads

Στην Παράγραφο 5.1.4 αναφέρθηκε ότι τα URBs τοποθετούνται στην κατάλληλη λίστα πριν την χρονοδρομολόγησή τους ανάλογα με τον HC που θα τα εξυπηρετήσει και το Schedule Division στο οποίο ανήκουν. Το λογισμικό τοποθετεί τα URBs του Schedule Division 0, δηλαδή εκείνα για τα οποία ισχύει (Pipe ID)% 2= 0, στο Schedule των frames για τα οποία ισχύει (Frame Number)% 2= 0, ενώ εκείνα του Schedule Division 1, δηλαδή εκείνα για τα οποία ισχύει (Pipe ID)% 2= 1, στο Schedule των frames για τα οποία ισχύει (Frame Number)% 2= 1. Έτσι οι λίστες URB του Schedule Division 0 εξυπηρετούνται από τα frames με άρτιο αριθμό, ενώ εκείνες του Schedule Division 1 εξυπηρετούνται από τα frames με περιττό αριθμό.

Σε κάθε Frame η διασύνδεση μεταξύ των Queue Heads του Schedule Division είναι σταθερή, όπως φαίνεται στο Σχήμα 5.2.1. Η διάταξη αυτή αναφέρεται ως Σκελετός των Queue Heads (*QH Skeleton*). Σημειώνεται ότι μια παρόμοια διάταξη των Queue Heads χρησιμοποιείται και στο Λειτουργικό Σύστημα Linux.



Σχήμα 5.2.1 Σκελετός των Queue Heads

Το Link Pointer του entry του Frame List που αντιστοιχεί στο τρέχον frame δείχνει στο πρώτο Isochronous TD του frame ή στο πρώτο Queue Head του σκελετού,

αν δεν υπάρχουν ισόχρονες αιτήσεις για αυτό το frame. Κάθε Queue Head δείχνει στο επόμενό του μέσω του Queue Head Link Pointer, και στην λίστα των TDs του (αν υπάρχουν) μέσω του Queue Element Link Pointer. Το τελευταίο Queue Head δείχνει NULL. Σημειώνεται ότι ακόμα και στην περίπτωση που το frame είναι κενό το Link Pointer του Frame List Entry θα δείχνει στο πρώτο QH και όχι NULL. Παρακάτω εξηγείται ποιές αιτήσεις μπαίνουν στην ουρά του κάθε Queue Head:

- Οι Interrupt αιτήσεις διαχωρίζονται ανάλογα με το Polling Interval. Με τον τρόπο αυτό επιτυγχάνεται η μείωση της πιθανότητας να μην εξυπηρετηθεί κάποια αίτηση λόγω σφάλματος σε κάποια αίτηση που βρίσκεται στην ίδια ουρά και προηγείται της αίτησης αυτής, δεδομένου ότι σε ένα υπολογιστικό σύστημα συνήθως δεν υπάρχουν Interrupt συσκευές με το ίδιο Interrupt Interval. Στις Interrupt συσκευές είναι κρίσιμο να μεταφερθούν χωρίς μεγάλη καθυστέρηση τα δεδομένα που εκκρεμούν. Επίσης το γεγονός ότι URB του ίδιου Pipe τοποθετούνται στην ίδια ουρά έχει ως αποτέλεσμα την τήρηση του πρωτοκόλλου Toggle Sequencing. Έτσι:

  - Στην ουρά του INTR_1 μπαίνουν οι αιτήσεις με Polling Interval μεγαλύτερο ή ίσο του 1 ms και μικρότερο των 2 ms.

  - Στην ουρά του INTR_2 μπαίνουν οι αιτήσεις με Polling Interval μεγαλύτερο ή ίσο των 2 ms και μικρότερο των 4 ms.

  - Στην ουρά του INTR_4 μπαίνουν οι αιτήσεις με Polling Interval μεγαλύτερο ή ίσο των 4 ms και μικρότερο των 8 ms.

  - Στην ουρά του INTR_8 μπαίνουν οι αιτήσεις με Polling Interval μεγαλύτερο ή ίσο των 8 ms και μικρότερο των 16 ms.

  - Στην ουρά του INTR_16 μπαίνουν οι αιτήσεις με Polling Interval μεγαλύτερο ή ίσο των 16 ms και μικρότερο των 32 ms.

  - Στην ουρά του INTR_32 μπαίνουν οι αιτήσεις με Polling Interval μεγαλύτερο ή ίσο των 32 ms και μικρότερο των 64 ms.

  - Στην ουρά του INTR_64 μπαίνουν οι αιτήσεις με Polling Interval μεγαλύτερο ή ίσο των 64 ms και μικρότερο των 128 ms.

  - Στην ουρά του INTR_128 μπαίνουν οι αιτήσεις με Polling Interval μεγαλύτερο ή ίσο των 128 ms και μικρότερο των 256 ms.

- Όλες οι Control αιτήσεις τοποθετούνται στην ουρά του Queue Head CONTROL.

- Οι Bulk αιτήσεις τοποθετούνται σε πέντε ουρές ως εξής: Στην ουρά του Queue Head BULK_1 τοποθετούνται μόνο τα Bulk URB του Pipe του πρώτου URB που θα τοποθετηθεί στην ουρά αυτή. Τα υπόλοιπα URB μπορούν να τοποθετηθούν σε μια από τις επόμενες ουρές. Το ίδιο ισχύει και για τις ουρές των Queue Heads BULK_2, BULK_3, και BULK_4. Δηλαδή κάθε μία από τις ουρές αυτές ανατίθεται σε ένα Bulk Pipe. Στην ουρά ANY_BULK τοποθετούνται οποιαδήποτε URBs με την προϋπόθεση ότι η διάρκειά τους δεν

ξεπερνάει το χρόνο του frame. Με τον τρόπο αυτό της δόμησης του Bulk Schedule μειώνεται η εξάρτηση της εκτέλεσης ενός TD από το αποτέλεσμα της εκτέλεσης κάποιου άλλου TD. Επίσης το γεγονός ότι URB του ίδιου Pipe τοποθετούνται στην ίδια ουρά έχει ως αποτέλεσμα την τήρηση του πρωτοκόλλου Toggle Sequencing.

5.2.4 Πρόσθεση Δομών Δεδομένων στο Schedule

Όταν ένα URB χρονοδρομολογείται σε ένα frame, το HCD δημιουργεί ένα Transfer Descriptor με βάση τις πληροφορίες του URB και το προσθέτει στο Schedule του frame στην κατάλληλη θέση. Το Link Pointer του προστιθέμενου TD πάντα τίθεται NULL. Εάν κάποιο επόμενο TD συνδεθεί με το TD αυτό, στο Link Pointer του TD θα γραφτεί η διεύθυνση του επόμενου TD. Η θέση στην οποία μπορεί να προστεθεί ένα TD μπορεί να είναι μια από τις επόμενες:

- Το TD είναι το πρώτο Isochronous TD του frame. Στην περίπτωση αυτή γράφεται στο Link Pointer του frame entry η διεύθυνση του TD (*append_td_to_frlist()*).

- Το TD είναι Isochronous αλλά δεν είναι το πρώτο Isochronous TD του frame. Στην περίπτωση αυτή γράφεται στο Link Pointer του προηγούμενου TD του frame η διεύθυνση του τρέχοντος TD (*append_td_to_td()*).

- Το TD είναι το πρώτο TD μιας Interrupt, Control, ή Bulk ουράς. Στην περίπτωση αυτή γράφεται στο Queue Element Link Pointer του QH η διεύθυνση του τρέχοντος TD (*append_td_to_qh()*).

- Το TD πρόκειται να προστεθεί σε μια Interrupt ή Bulk ουρά, αλλά δεν είναι το πρώτο TD της ουράς. Στην περίπτωση αυτή γράφεται στο Link Pointer του τελευταίου TD της ουράς (το οποίο δεν είναι αναγκαστικά το τελευταίο TD που προστέθηκε στο Schedule του frame) η διεύθυνση του τρέχοντας TD (*append_td_to_intr_td()*).

- Το TD πρόκειται να προστεθεί στην Control ουρά, αλλά δεν είναι το πρώτο TD της ουράς. Στην περίπτωση αυτή γράφεται στο Link Pointer του τελευταίου TD της ουράς (το οποίο είναι και το τελευταίο TD που προστέθηκε στο Schedule του frame) η διεύθυνση του τρέχοντας TD (*append_td_to_td()*).

Υπενθυμίζεται ότι το λογισμικό διατηρεί έναν μετρητή των χρησιμοποιημένων TDs κάθε frame έτσι ώστε να μπορεί να βρει την διεύθυνση του τρέοντος και του τελευταίου TD που προστέθηκε στο Schedule. Επιπλέον διατηρεί την διεύθυνση του τελευταίου TD που προστέθηκε σε κάθε Interrupt ή Bulk ουρά.

Σε ότι αφορά τα Queue Heads:

- Κάθε Queue Head περιέχει στο Queue Head Link Pointer την διεύθυνση του επόμενου QH του σκελετού (*append_qh_to_qh_hor*), εκτός από το τελευταίο QH που δείχνει NULL.

- Εάν δεν υπάρχει κανένα Isochronous TD στο frame τότε γράφεται στο Link Pointer του frame entry η διεύθυνση του πρώτου QH του σκελετού (*append_qh_to_frlist()*).

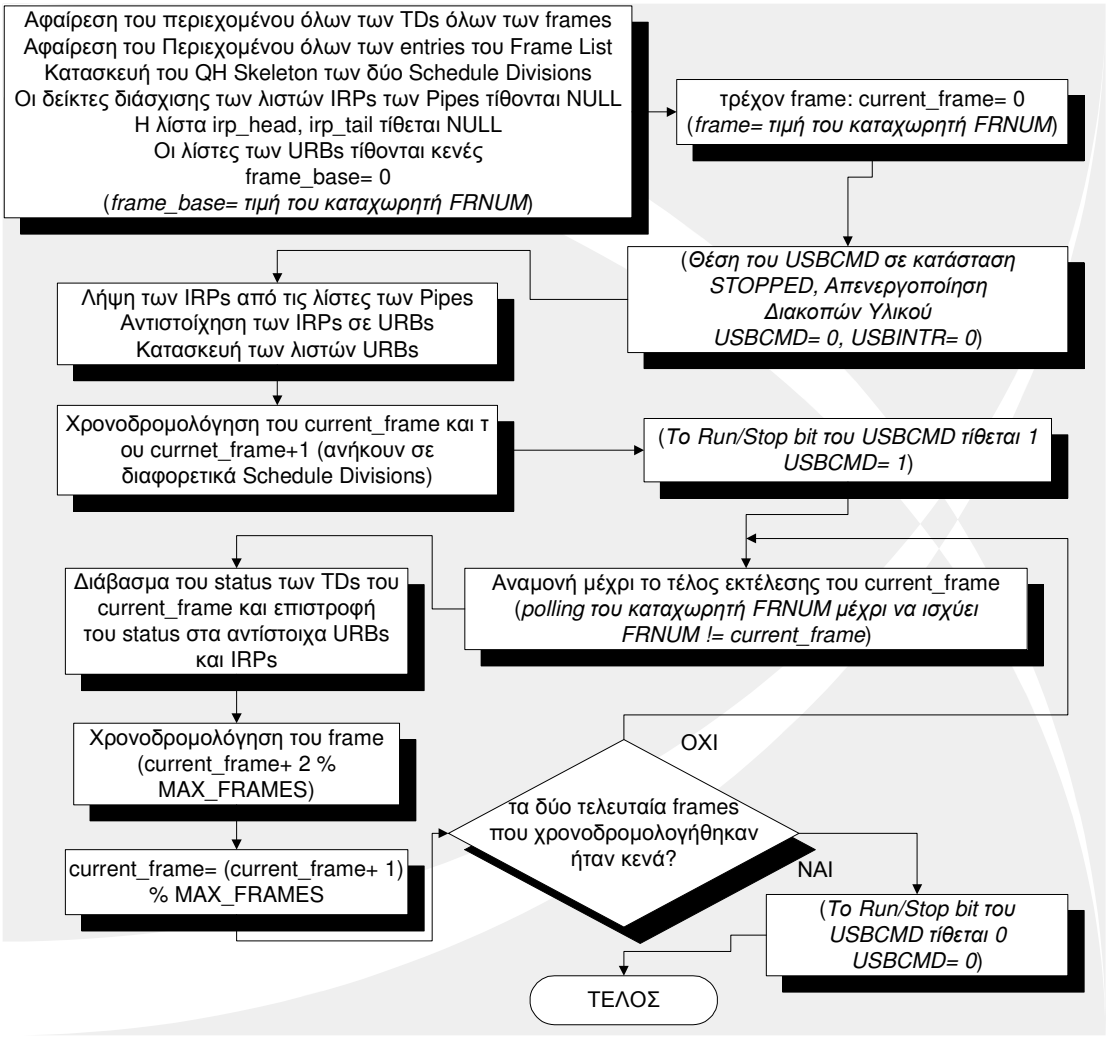
5.3 Υλοποίηση της Χρονοδρομολόγησης των Αιτήσεων Μεταφοράς Δεδομένων

Στις παραγράφους που ακολουθούν παρουσιάζονται οι αλγόριθμοι χρονοδρομολόγησης των αιτήσεων μεταφοράς δεδομένων σε έναν Host Controller. Οι αλγόριθμοι αυτοί είναι σύμφωνοι με τις προδιαγραφές του Universal Serial Bus 1.1 και του Universal Host Controller Interface. Έπειτα παρουσιάζονται οι αλγόριθμοι ανάγνωσης του status κάθε transaction ενός frame και επιστροφής του στα IRP στο οποίο ανήκει.

5.3.1 Χρονοδρομολόγηση Συσκευής

Στο Σχήμα 5.3.1 παρουσιάζεται ο αλγόριθμος που υλοποιήθηκε για την χρονοδρομολόγηση των αιτήσεων μεταφοράς δεδομένων σε έναν Host Controller. Ο αλγόριθμος αυτός χρησιμοποιεί διαδοχικά frames για την χρονοδρομολόγηση των αιτήσεων, μέχρι να εκτελεστούν όλα τα εκκρεμή URB. Στον αλγόριθμο παρουσιάζονται οι ενέργειες που πραγματοποιούνται για την χρονοδρομολόγηση των αιτήσεων μεταφοράς δεδομένων σε έναν εικονικό Host Controller, ο οποίος υλοποιείται μέσω ενός συμπληρωματικού λογισμικού κατά τον έλεγχο της ορθότητας του Scheduling και προσωμειώνει τις βασικές λειτουργίες ενός UHCI Host Controller (βλέπε Ενότητα 5.5). Στις παρενθέσεις και με πλάγια γραφή δείχνονται οι τροποποιήσεις στις ενέργειες του αλγορίθμου έτσι ώστε να οδηγεί έναν πραγματικό Host Controller.

Αρχικά, προετοιμάζονται οι λίστες των IRPs και URBs και οι δομές δεδομένων του Scheduling για την επεξεργασία τους. Η μεταβλητή frame_base, η οποία χρησιμοποιείται για την χρονοδρομολόγηση των Isochronous αιτήσεων τίθεται 0 (*ίση με τον αριθμό του τρέχοντος frame*). Στη συνέχεια η τιμή του τρέχοντος frame τίθεται 0 (*ίση με το περιεχόμενο του καταχωρητή FRNUM*). Το HCD δημιουργεί τις λίστες των URBs χρησιμοποιώντας τους αλγορίθμους των σχημάτων 5.1.1 έως 5.1.4. Το HCD διαβάζει τα περιεχόμενα των λιστών, χρονοδρομολογεί τα δύο πρώτα frames και αρχίζει την εικονική εκτέλεση του Schedule (*θέτει κατάλληλα τον καταχωρητή USBCMD έτσι ώστε να αρχίσει την εκτέλεση του Schedule*). Περιμένει την εκτέλεση του τρέχοντος frame (*περιμένοντας την αλλαγή του περιεχομένου του καταχωρητή FRNUM*). Όταν αυτό συμβεί, διαβάζει το status της εκτέλεσης των TDs και το επιστρέφει στα αντίστοιχα URBs και τα IRPs. Με βάση το status κάθε TD οι λίστες των URBs ανασυντάσσονται όπως θα φανεί σε επόμενη παράγραφο Στη συνέχεια χρονοδρομολογεί το μεθεπόμενο frame (το οποίο ανήκει στο ίδιο Schedule Division) με το νέο περιεχόμενο των λιστών. Η τιμή του τρέχοντος frame αυξάνεται κατά ένα έτσι ώστε να επαναλάβει τις ίδιες ενέργειες για το επόμενο frame (το οποίο έχει ήδη χρονοδρομολογηθεί). Όταν τα δύο τελευταία frames που χρονοδρομολογήθηκαν είναι κενά η επεξεργασία σταματάει (*θέτοντας κατάλληλα τον USBCMD*). Ο έλεγχος των δύο τελευταίων frames, και όχι μόνο του τελευταίου, είναι απαραίτητος γιατί δύο διαδοχικά frames διαχειρίζονται διαφορετικές λίστες URBs (διαφορετικό Schedule Division).



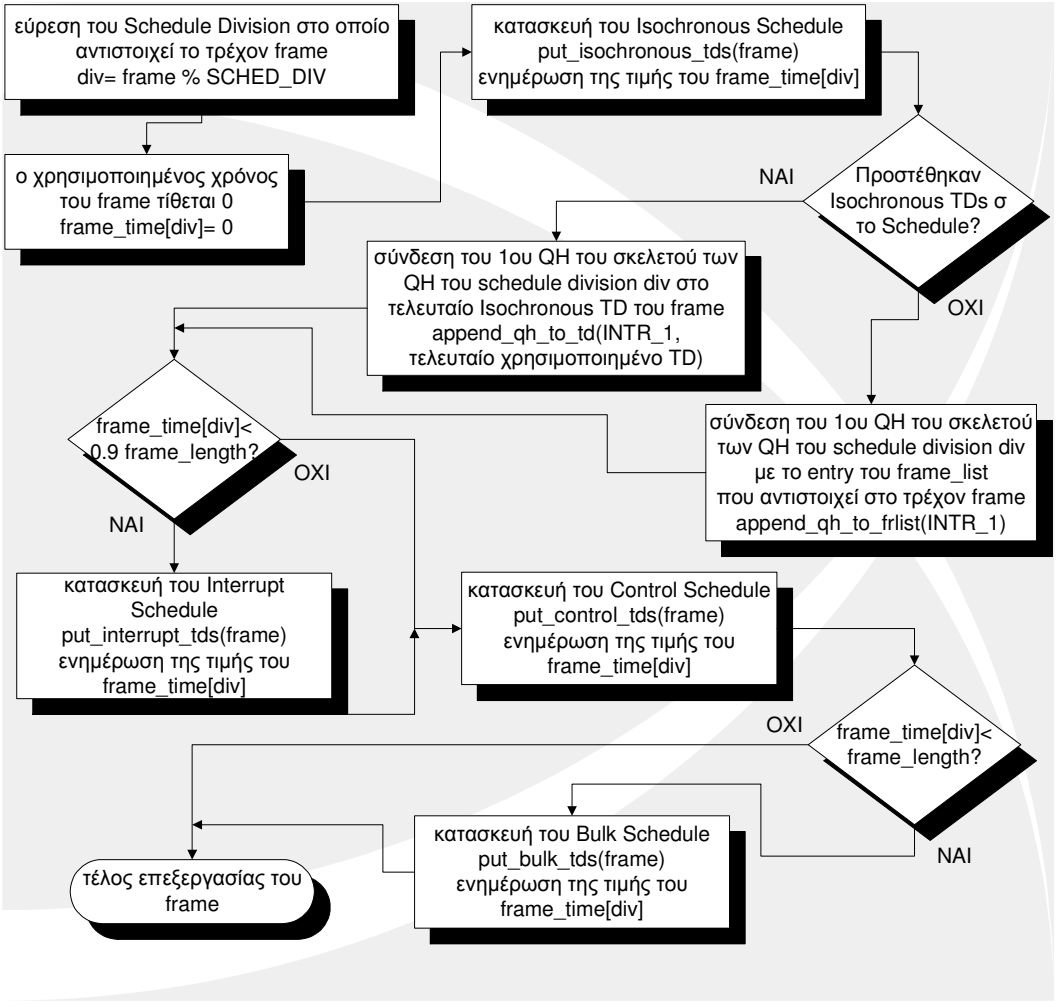
Σχήμα 5.3.1: Αλγόριθμος Χρονοδρομολόγησης των Αιτήσεων Μεταφοράς Δεδομένων σε έναν Host Controller

Στο σημείο αυτό είναι εμφανής ο λόγος για τον οποίο είναι απαραίτητη η διαίρεση των αιτήσεων μεταφοράς δεδομένων και του Schedule σε Schedule Divisions. Το HCD χρονοδρομολογεί ένα frame και στη συνέχεια περιμένει να εκτλεστεί έτσι ώστε να λάβει το status των transactions, να ενημερώσει τις λίστες των URBs και να χρονοδρομολογήσει το επόμενο frame που ανήκει στο ίδιο Schedule Division (δύο frames μετά). Αν υποτεθεί ότι δεν υπάρχει διαχωρισμός σε Disivions, το HCD θα πρέπει να χρονοδρομολογήσει το αμέσως επόμενο frame εκείνου που μόλις ολοκληρώθηκε. Όμως, η παραγωγή των frames από τον HC είναι συνεχής και το HCD πιθανότατα δεν θα έχει προλάβει να κατασκευάσει το Schedule του επόμενου frame προτού ο HC να διαβάσει στις δομές δεδομένων του frame, με αποτέλεσμα την λανθασμένη εκτέλεσή του. Για παράδειγμα, μπορεί το HCD να μην έχει προλάβει να προσθέσει TDs στο schedule του frame τη στιγμή που ο HC διαβάζει το entry του frame list που αντιστοιχεί στο τρέχον frame, ενώ υπάρχουν εκκρεμή URBs προς εκτέλεση στο τρέχον frame, με αποτέλεσμα το αντίστοιχο entry του frame list να δείχνει τη στιγμή εκείνη NULL και ο HC να θεωρεί λανθασμένα ότι το τρέχον frame είναι κενό. Με την χρήση όμως των Schedule Divisions, το επόμενο frame του τρέχοντος θα είναι πάντα χρονοδρομολογημένο πριν την ολοκλήρωση του

τρέχοντος frame, δίνοντας μια έμμεση λύση στο πρόβλημα του συγχρονισμού του υλικού με τον λογισμικό.

5.3.2 Χρονοδρομολόγηση Πλαισίου (Frame)

Ο αλγόριθμος του Σχήματος 5.3.2 χρησιμοποιείται για την χρονοδρομολόγηση ενός frame με τις αιτήσεις μεταφοράς δεδομένων του αντίστοιχου Schedule Division. Ο αποδιδόμενος χρόνος και η προτεραιότητα για τις αιτήσεις κάθε τύπου μεταφοράς δεδομένων είναι σύμφωνα με τις προδιαγραφές του USB 1.1. Δηλαδή, προτεραιότητα δίνεται πρώτα στις Isochronous μεταφορές και έπειτα στις Interrupt. Για τις παραπάνω μεταφορές μπορεί να εκχωρηθεί το πολύ το 90 % του χρόνου του frame. Ο υπόλοιπος χρόνος του frame μπορεί να χρησιμοποιηθεί για Control μεταφορές. Εάν παραμείνει διαθέσιμος χρόνος σε ένα frame, μπορεί να χρησιμοποηθεί για Bulk μεταφορές. Επιπλέον ο αλγόριθμος αυτός συνδέει το κατάλληλο αντικείμενο του Schedule (πρώτο isochronous TD ή πρώτο QH) με το τρέχον entry του frame list. Στο array frame_time[] αποθηκέυεται ο χρησιμοποιημένος χρόνος του frame για κάθε Schedule Division.



Σχήμα 5.3.2: Αλγόριθμος Χρονοδρομολόγησης Πλαισίου

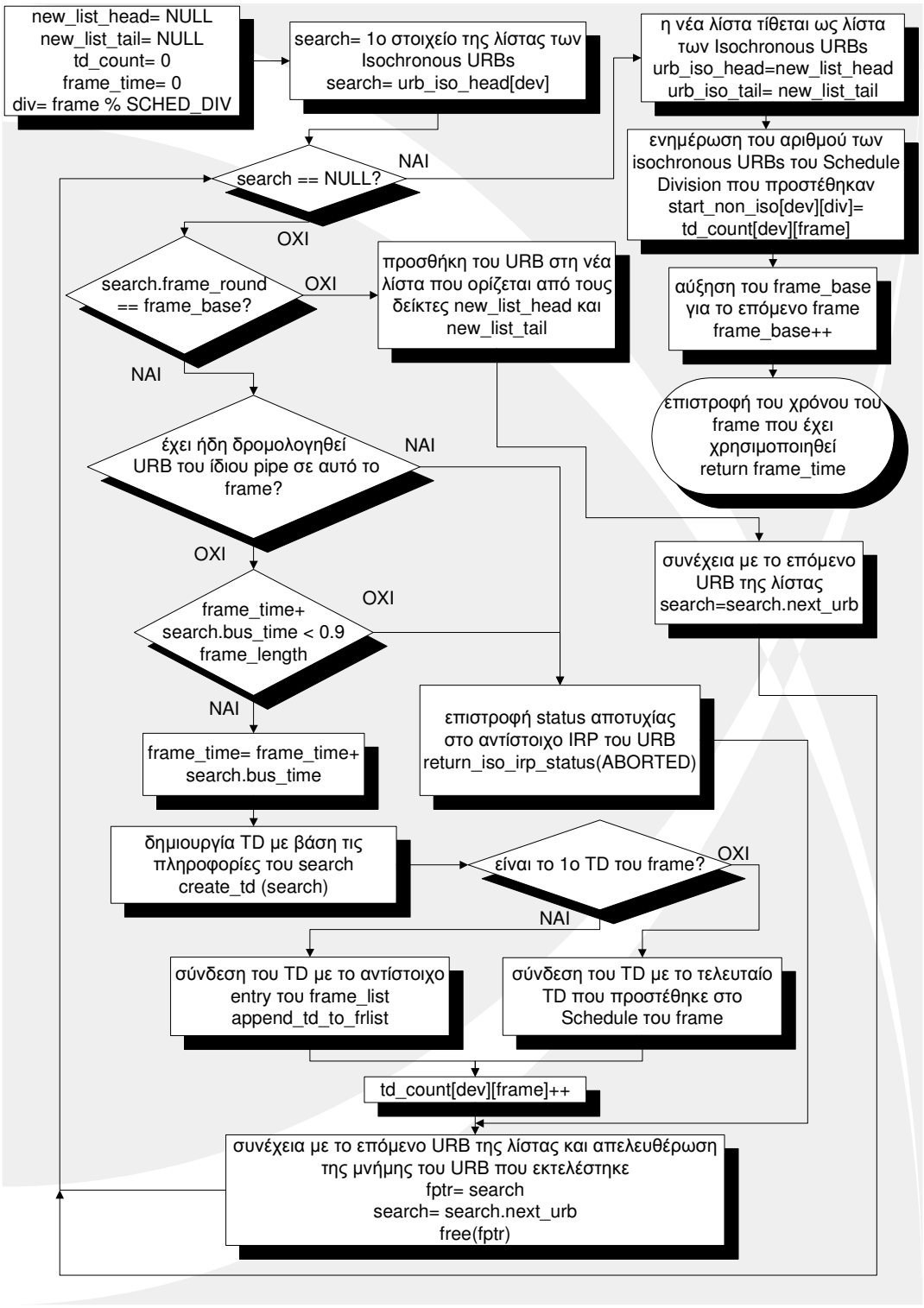
Στις επόμενες υποπαραγράφους περιγράφονται οι αλγόριθμοι χρονοδρομολόγησης των Isochronous, Interrupt, Control, και Bulk αιτήσεων σε ένα frame.

5.3.2.1 Χρονοδρομολόγηση των Isochronous Αιτήσεων

Ο αλγόριθμος του Σχήματος 5.3.3 χρησιμοποιείται για την χρονοδρομολόγηση των Isochronous αιτήσεων σε ένα frame.

Ο δείκτης search διατρέχει την λίστα των Isochronous URBs του HC μέχρι το τέλος της. Το πεδίο frame_round των Isochronous URBs δείχνει το επιθυμητό frame εκτέλεσης του transaction που αντιπροσωπεύει το USB. Αν αυτό είναι ίσο με το τρέχον frame, το οποίο δηλώνεται από την μεταβλητή frame_base τότε το URB θα επεξεργαστεί, αλλιώς όχι. Η τιμή της μεταβλητής frame_base και του πεδίου frame_round των URB δεν περιορίζεται μέχρι την τιμή 1023 αλλά λαμβάνει όλες τις ακέραιες τιμές που μπορεί να λάβει ένας interger στο σύστημα, έτσι ώστε να μην υπάρχει σημαντικός περιορισμός στον αριθμό των μελλοντικών frames στα οποία να μπορούν να χρονοδρομολογηθούν isochronous αιτήσεις.

Κάθε URB που επεξεργάζεται ελέγχεται για το αν μπορεί να χωρέσει στο frame και για το αν έχει ήδη δρομολογηθεί ένα URB του ίδιου pipe στο τρέχον frame. Σε κανονικές συνθήκες, το USBD είναι υπεύθυνο για την τήρηση των παραπάνω συνθηκών. Για κάθε ενδεχόμενο το HCD ελέγχει την ισχύ τους. Αν οι συνθήκες πληρούνται, τότε δημιουργείται το αντίστοιχο TD το οποίο προστίθεται στην κατάλληλη θέση του schedule, αλλιώς το URB αποσύρεται και επιστρέφεται status αποτυχίας στο αντίστοιχο IRP. Στο τέλος της επεξεργασίας στην λίστα των Isochronous URBs παραμένουν τα URBs για τα οποία το επιθυμητό frame εκτέλεσης είναι διάφορο από το τρέχον frame. Τα υπόλοιπα αφαιρούνται, είτε προστέθηκαν στο Schedule είτε όχι. Το HCD επιστρέφει στην καλούσα συνάρτηση τον χρόνο του frame που έχει χρησιμοποιηθεί, ενώ αποθηκεύει τον αριθμό των Isochronous TDs που χρησιμοποιήθηκαν.

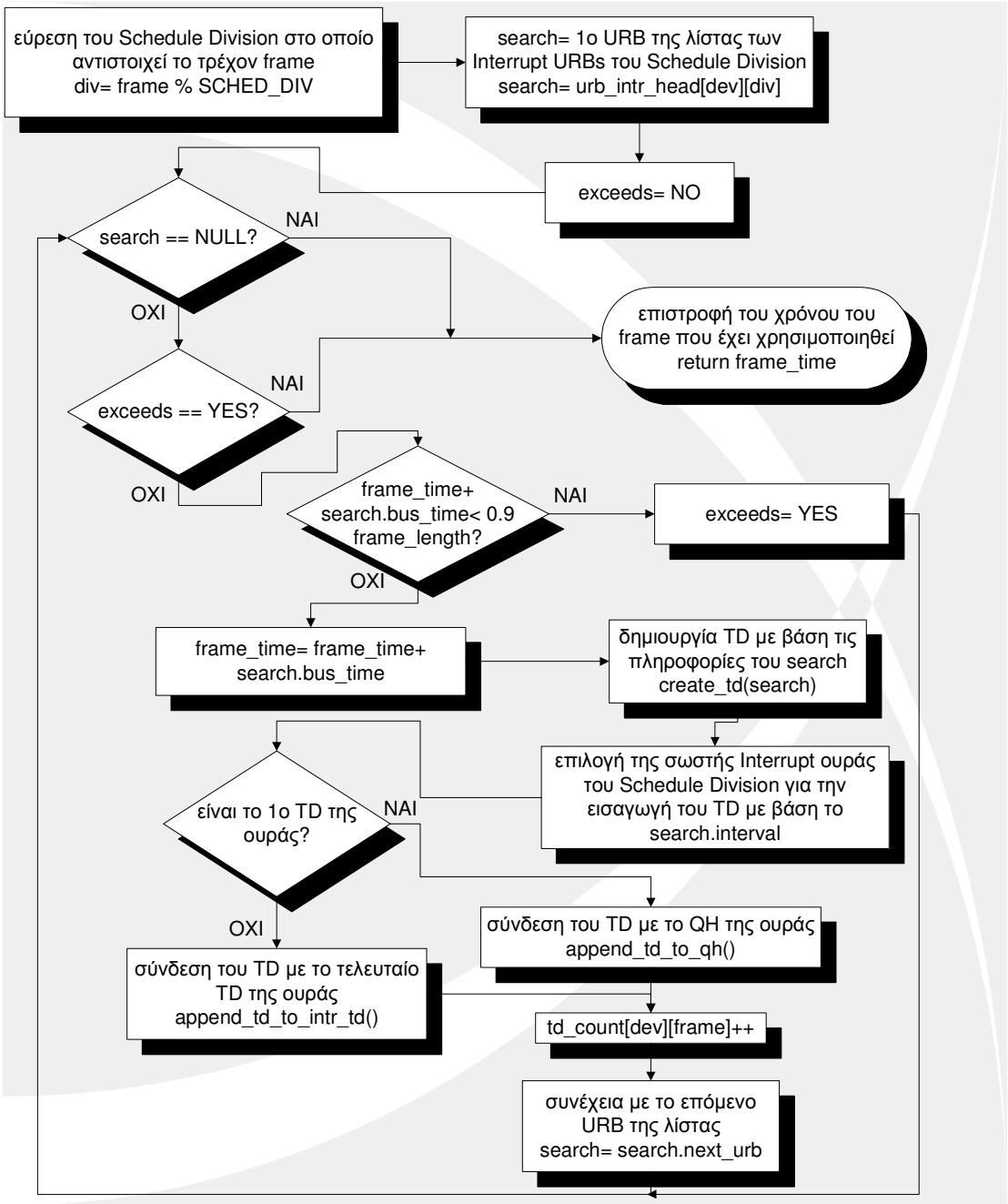


Σχήμα 5.3.3: Αλγόριθμος Χρονοδρομολόγησης των Isochronous Αιτήσεων

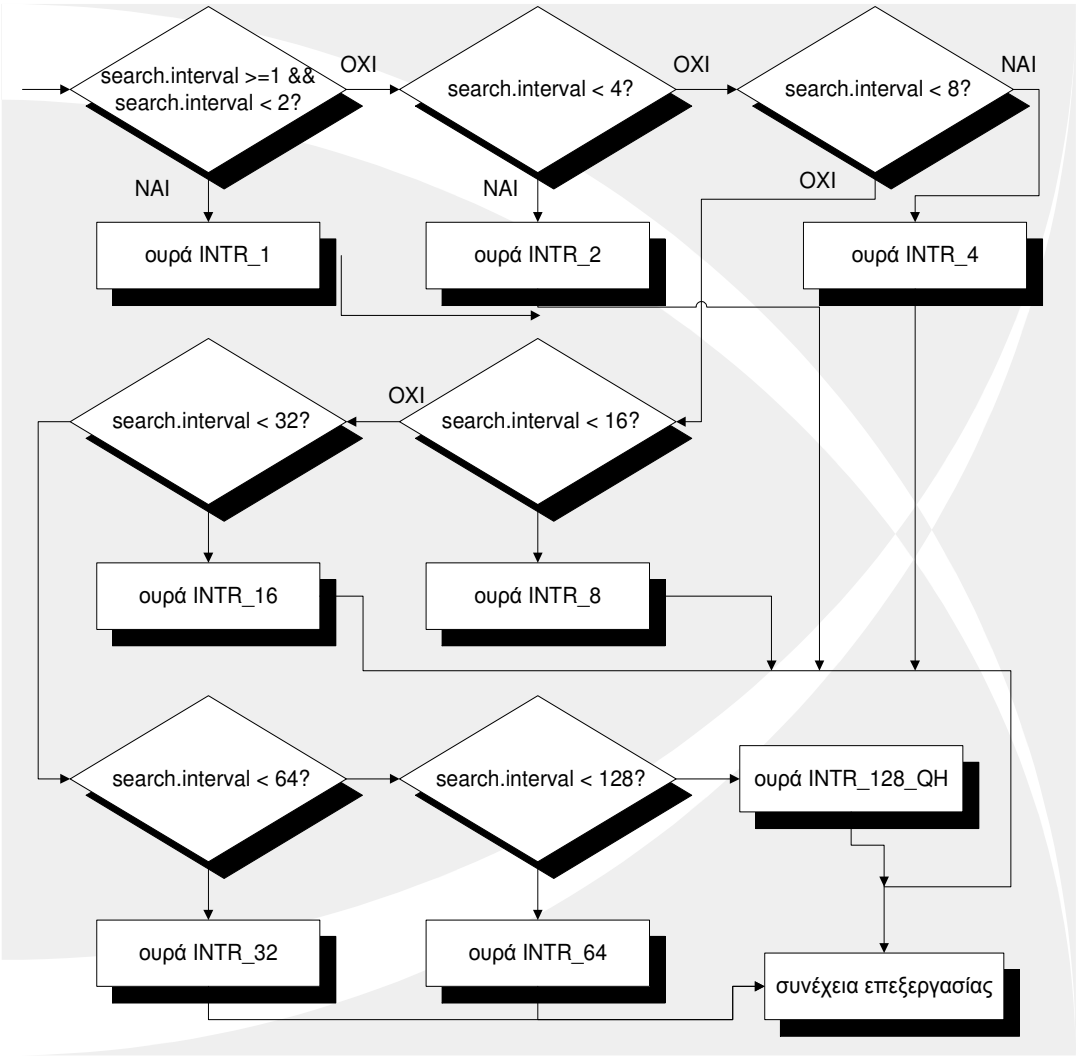
5.3.2.2 Χρονοδρομολόγηση των Interrupt Αιτήσεων

Ο αλγόριθμος του Σχήματος 5.3.4 χρησιμοποιείται για την χρονοδρομολόγηση των Interrupt αιτήσεων σε ένα frame. Ο δείκτης search διατρέχει την λίστα των Interrupt URBs του Schedule Division στο οποίο αντιστοιχεί το τρέχον frame και του

συγκεκριμένου HC. Αν ο χρόνος διαύλου του URB είναι τέτοιος έτσι ώστε να χωράει στο τρέχον frame, δηλαδή στον χρόνο που έχει εκχωρηθεί για περιοδικές αιτήσεις μεταφοράς δεδομένων, το αντίστοιχο TD δημιουργείται με βάση τις πληροφορίες του URB και προστίθεται στην κατάλληλη ουρά του Schedule. Η επιλογή της ουράς γίνεται με βάση τον αλγόριθμο του Σχήματος 5.3.5. Αν το URB δεν χωράει, τότε η διαδικασία ολοκληρώνεται χωρίς να ελέγχεται αν κάποιο άλλο URB δύναται να χωρέσει στον υπολοιπόμενο διαθέσιμο χρόνο. Αυτό γίνεται για να μην παραβιαστούν οι κανόνες του πρωτοκόλλου Toggle Sequencing.



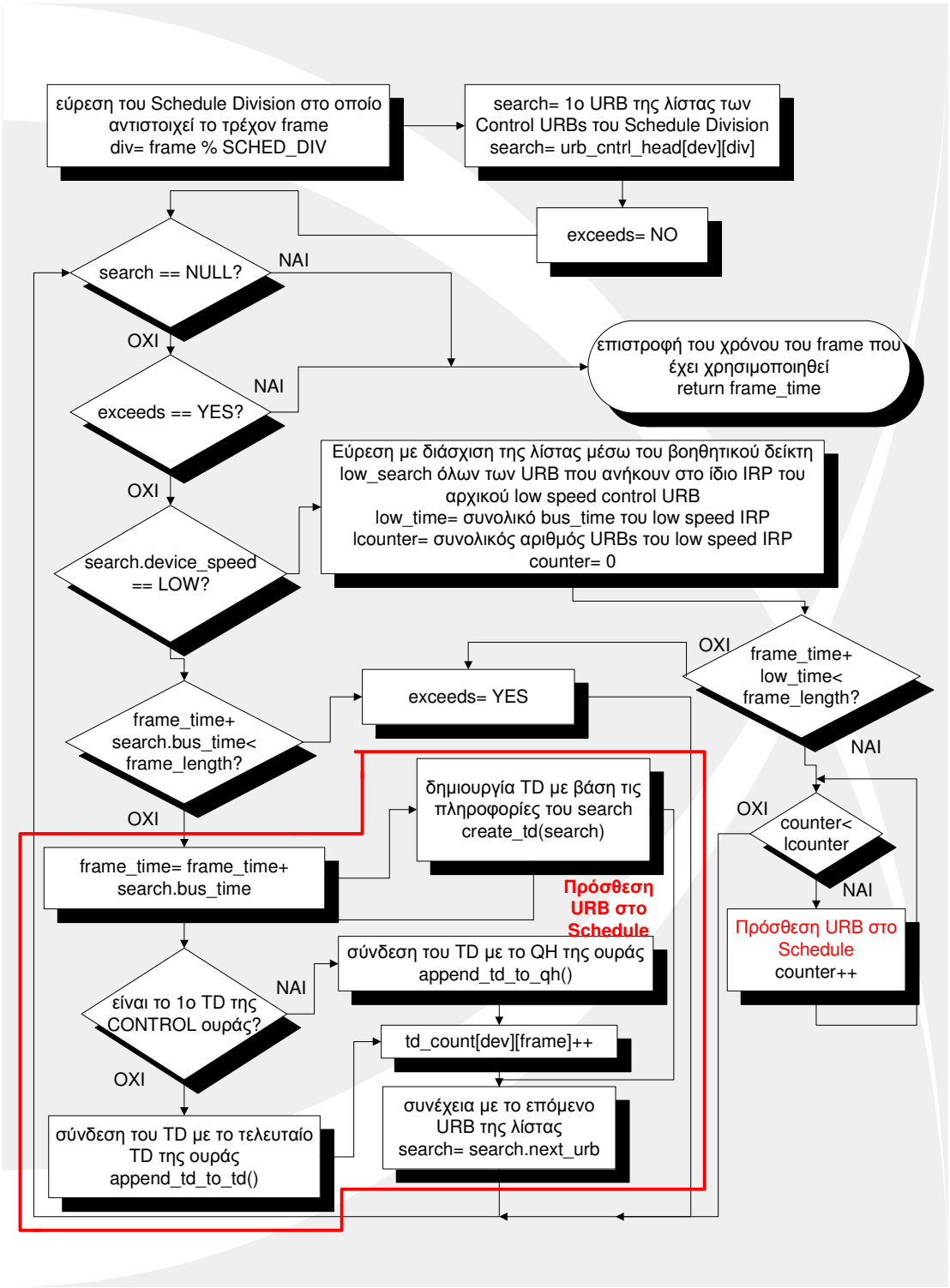
Σχήμα 5.3.4: Αλγόριθμος Χρονοδρομολόγησης των Interrupt Αιτήσεων



Σχήμα 5.3.5: Αλγόριθμος Επιλογής της Αντίστοιχης Interrupt Ουράς μιας Interrupt Αίτησης

5.3.2.3 Χρονοδρομολόγηση των Control Αιτήσεων

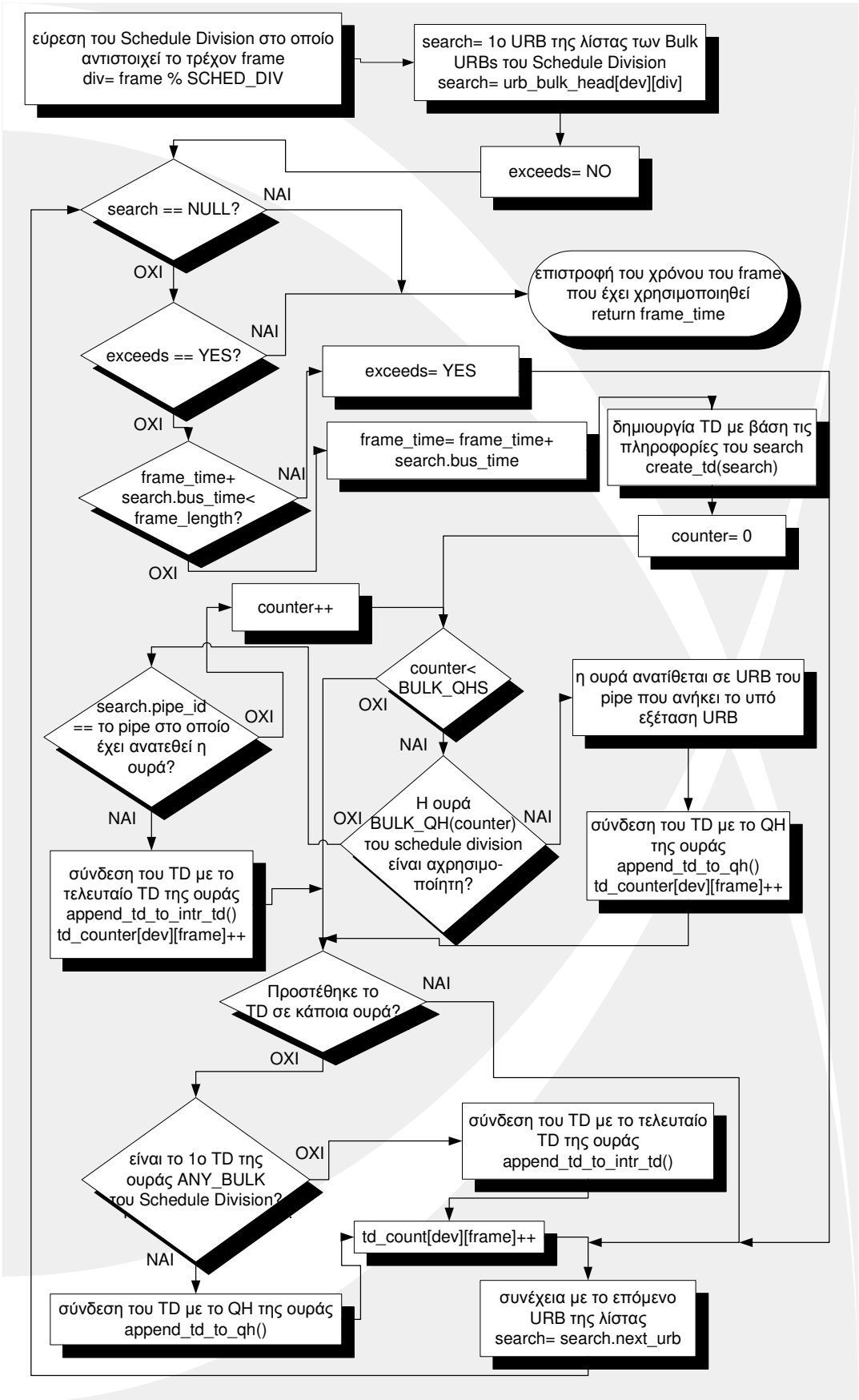
Ο αλγόριθμος του Σχήματος 5.3.6 χρησιμοποιείται για την χρονοδρομολόγηση των Control αιτήσεων σε ένα frame. Ο αλγόριθμος αυτός επεξεργάζεται τις Full-speed Control αιτήσεις με όμοιο τρόπο με τον οποίο επεξεργάζεται ο αλγόριθμος του σχήματος 5.3.4 τις Interrupt αιτήσεις, με μόνη διαφορά ότι όλες οι αιτήσεις συνδέονται στην ουρά CONTROL. Το UHCI απαιτεί οποιαδήποτε Low-speed Control μεταφορά να ολοκληρώνεται στο ίδιο frame στο οποίο ξεκινάει. Δηλαδή, όλα τα URB τα οποία συνθέτουν μια Low-speed Control μεταφορά να μεταφράζονται στα αντίστοιχα TD και να τοποθετούνται στο Schedule ενός μόνο frame. Για τον λόγο αυτό, σύμφωνα με τον αλγόριθμο, όταν το search δείχνει σε κάποιο Low-speed Control URB, ένας βοηθητικός δείκτης διατρέχει την λίστα από το σημείο στο οποίο δείχνει ο δείκτης search, και βρίσκει το σύνολο των URB που συνθέτουν την Low-speed Control μεταφορά. Στη συνέχεια υπολογίζεται ο συνολικός χρόνος διαύλου της Low-speed Control μεταφοράς. Εάν όλα τα URB που τη συνθέτουν χωράνε στον διαθέσιμο χρόνο τότε προσθέτονται στην ουρά, αλλιώς η επεξεργασία ολοκληρώνεται επιστρέφοντας στην καλούσα συνάρτηση.



Σχήμα 5.3.6: Αλγόριθμος Χρονοδρομολόγησης των Control Αιτήσεων

5.3.2.4 Χρονοδρομολόγηση των Bulk Αιτήσεων

Ο αλγόριθμος του Σχήματος 5.3.7 χρησιμοποιείται για την χρονοδρομολόγηση των Bulk αιτήσεων σε ένα frame.



Σχήμα 5.3.7: Αλγόριθμος Χρονοδρομολόγησης των Bulk Αιτήσεων

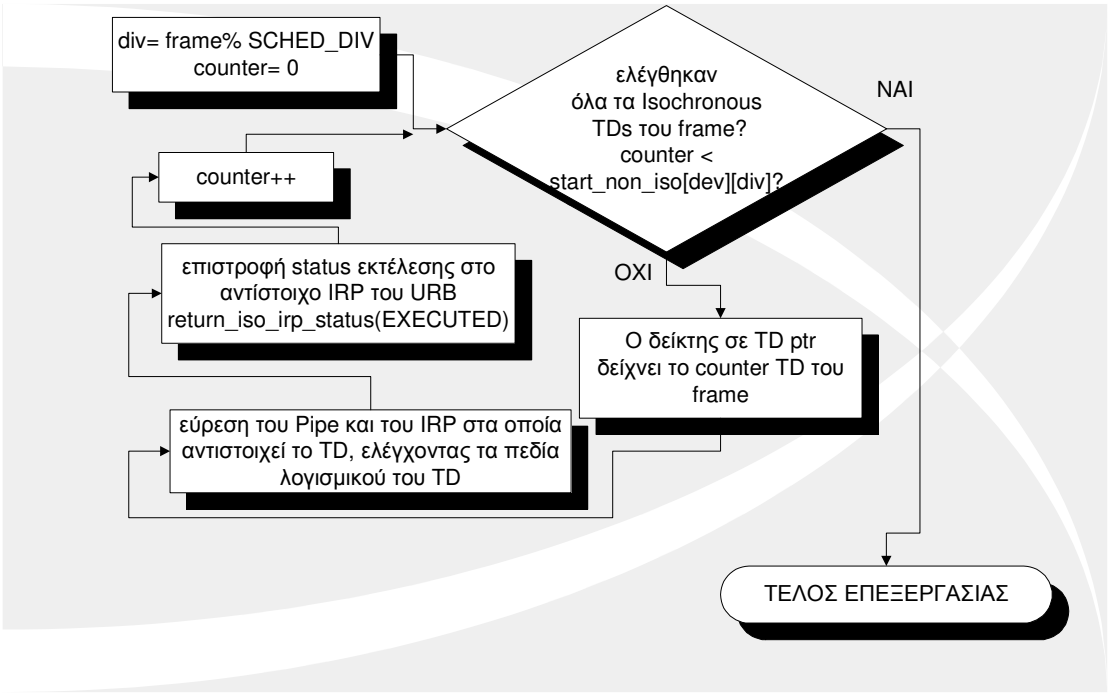
Η λογική του αλγορίθμου αυτού είναι όμοια με αυτή των αλγορίθμων των Interrupt αιτήσεων. Η μόνη διαφορά είναι ο τρόπος με τον οποίο επιλέγεται η ουρά εισαγωγής του νέου TD. Όπως έχει ήδη αναφερθεί, οι τέσσερις πρώτες Bulk ουρές χρησιμοποιούνται για την εξυπηρέτηση των αιτήσεων ενός μόνο Pipe η κάθε μία. Το Pipe στο οποίο ανατίθεται κάθε ουρά δεν είναι προκαθορισμένο, αλλά εξαρτάται από τα Pipes στα οποία ανήκουν τα URBs που συναντώνται διατρέχοντας την λίστα των URBs. Έτσι η πρώτη ουρά ανατίθεται στα URBs του Pipe στο οποίο ανήκει το πρώτο Bulk URB που προστίθεται στο Schedule. Η δεύτερη ουρά ανατίθεται στα URBs του Pipe στο οποίο ανήκει το πρώτο Bulk URB το οποίο δεν ανήκει στο Pipe της πρώτης ουράς κ.ο.κ. Όσα Bulk URBs χωράνε στον διαθέσιμο χρόνο του frame και δεν έχουν «προλάβει» να καταλάβουν κάποια από τις τέσσερις πρώτες ουρές τοποθετούνται στην ουρά ANY_BULK.

5.3.3 Επιστροφή του Status των Transactions

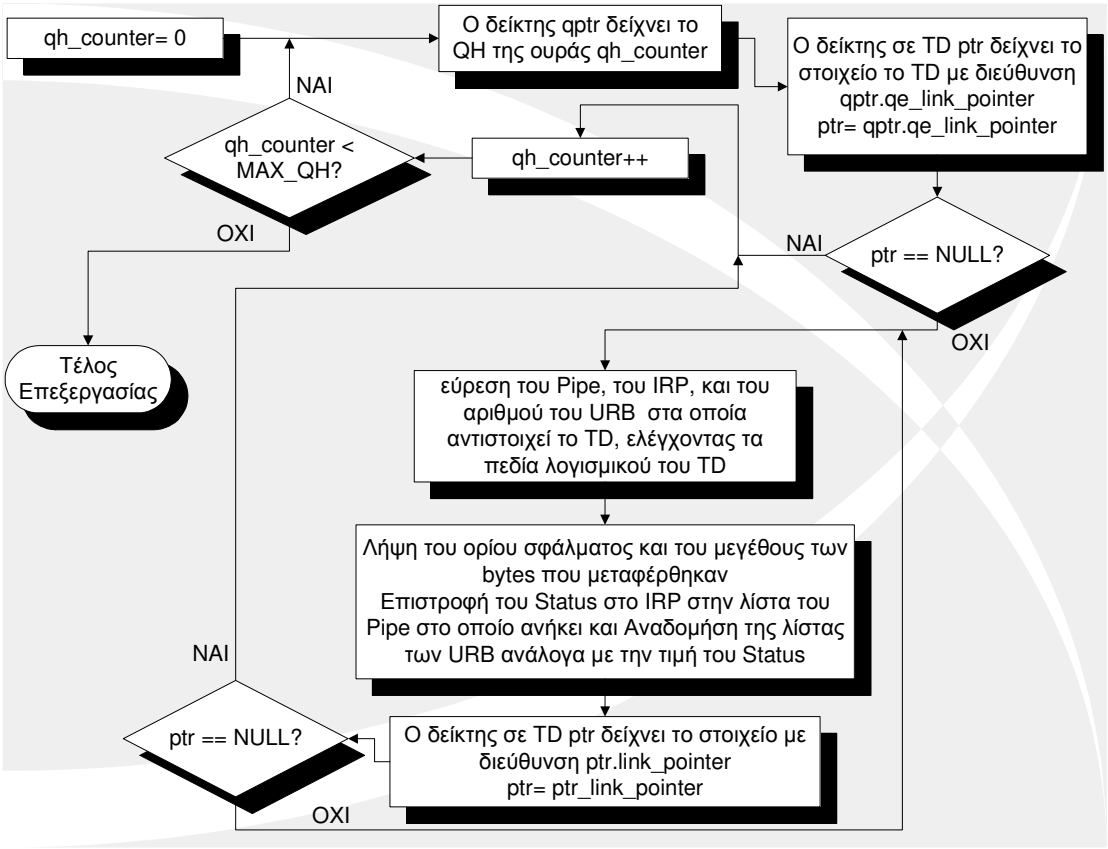
Μετά την εκτέλεση της λίστας των transactions του τρέχοντος frame, το HCD διασχίζει τις δομές δεδομένων που συνθέτουν το Schedule του frame για να διαβαστεί το status κάθε transaction, ελέγχοντας το πεδίο TD Control & Status του Trasfer Descriptor που αντιπροσωπεύει κάθε transaction. Το HCD ελέγχει το status κάθε transaction, το επιστρέφει στο αντίστοιχο IRP στο οποίο ανήκει το URB από το οποίο δομήθηκε το TD του transaction, και αναδομεί τις λίστες των URBs της συσκευής και του συγκεκριμένου schedule Division, για την χρονοδρομολόγηση του επόμενου frame του Schedule Division.

Σε ό,τι αφορά τα Isochronous URBs, η διαδικασία είναι ιδιαίτερα απλή, όπως φαίνεται στον αλγόριθμο του Σχήματος 5.3.8. Το HCD διασχίζει όλα τα Isochronous TDs του frame και για κάθε ένα επιστρέφει στο αντίστοιχο IRP το status επιτυχίας (*return_iso_irp_status(EXECUTED)*). Αυτό γίνεται χρησιμοποιώντας τις πρόσθετες πληροφορίες του TD για να διασχίζει την λίστα των IRPs του Pipe, να βρει το συγκεκριμένο IRP, και να εγγράψει το status στο κατάλληλο πεδίο του IRP. Υπενθυμίζεται ότι στα Isochronous TD δεν γίνεται ανίχνευση σφαλμάτων από το υλικό, οπότε το λογισμικό δεν μπορεί να πκληροφορηθεί για το αν το TD εκτελέστηκε όντως με επιτυχία. Η λίστα των Isochronous URBs δεν αναδομείται, αφού κάθε URB που προστίθεται στην λίστα αφαιρείται την στιγμή της προσθήκης του.

Σε ό,τι αφορά τα Non-Isochronous TDs, το λογισμικό διασχίζει τις ουρές του Non-Isochronous Schedule για να διαβάσει το Status κάθε TD, όπως φαίνεται στον αλγόριθμο του Σχήματος 5.3.9. Το HCD διαβάζει τις πρόσθετες πληροφορίες του TD (πεδία λογισμικού) έτσι ώστε να βρει την ταυτότητα του Pipe και του IRP που αντιστοιχεί στο TD, και τον αριθμό του URB. Επίσης διαβάζει την τιμή του ορίου σφαλμάτων του TD και το μέγεθος των bytes που μεταφέρθηκαν στο transaction. Έπειτα, το HCD διαβάζει το status που έχει γραφτεί από το υλικό στο πεδίο TD Control & Status, και ανάλογα με την τιμή του κάνει τις κατάλληλες επεξεργασίες:



Σχήμα 5.3.8: Αλγόριθμος Επιστροφής του status των Isochronous TDs



Σχήμα 5.3.9: Αλγόριθμος Διάσχισης του Schedule για την Ανάγνωση του Status των Non-Isochronous Transactions

- Εάν αναφέρεται λήψη χειραψίας NAK, το HCD ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB θα ξαναεκτελεστεί και επιστρέφει το αντίστοιχο status (return_irp_status (*TO_BE_RETRIED)*). Το αντίστοιχο URB παραμένει στην λίστα του με το ίδιο όριο σφαλμάτων (Η λήψη χειραψίας NAK δεν θεωρείται κατάσταση σφάλματος)..

- Εάν αναφέρεται σφάλμα BitStuff, CRC/Timeout, ή Buffer Error και το όριο σφαλμάτων δεν έχει γίνει μηδέν (*NOT_CRITICAL_ERROR*), το HCD ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB θα ξαναεκτελεστεί και επιστρέφει το αντίστοιχο status (return_irp_status (*TO_BE_RETRIED)*). Το αντίστοιχο URB παραμένει στην λίστα του με την νέα τιμή του ορίου σφάλματος (μειωμένη κατά ένα).

- Εάν αναφέρεται σφάλμα Stalled, Babble Detected, ή το όριο σφάλματος του TD ξεπεραστεί (*CRITICAL_ERROR),* το HCD ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB αποσύρεται και επιστρέφει το αντίστοιχο status (*return_irp_status (ABORTED)*). Επίσης επιστρέφει σε όλα τα IRP του ίδιου Pipe status αποτυχίας λόγω σφάλματος σε άλλο IRP του ίδιου Pipe (*return_irp_status (ABORTED & FROM_OTHER)*). Το HCD αφαιρεί από την λίστα των URBs όλα τα URB που αντιστοιχούν στο Pipe του αποτυχημένου URB.

- Εάν δεν αναφέρεται κάποιο σφάλμα, το TD παραμένει Active, και δεν ανήκει σε Pipe στο οποίο έχει αναφερθεί κρίσιμο σφάλμα σημαίνει ότι το TD δεν εκτελέστηκε λόγω σφάλματος σε κάποιο προηγούμενο TD της ίδιας ουράς, ή λόγω ανίχνευσης Babble σε προηγούμενο TD του frame. Το HCD ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB θα εκτελεστεί ξανά λόγω σφάλματος σε άλλο URB (*return_irp_status (TO_BE_RETRIED & FROM_OTHER)*). Το αντίστοιχο URB παραμένει στην λίστα του. Εάν το TD ανήκει σε Pipe στο οποίο έχει αναφερθεί κρίσιμο σφάλμα το HCD δεν πραγματοποιεί κάποια επεξεργασία.

- Εάν δεν αναφέρεται κάποιο σφάλμα και το TD είναι Inactive, σημαίνει ότι το TD εκτελέστηκε επιτυχώς. Το HCD ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB εκτελέστηκε επιτυχώς επιστρέφοντας τον αριθμό των bytes που μεταφέρθηκαν (*return_irp_status(EXECUTED)*). Το αντίστοιχο URB αφαιρείται από την λίστα του.

Μετά το τέλος όλων των επεξεργασιών του λογισμικού, τα επιχυχή IRPs θα είναι εκείνα για τα οποία έχει επιστραφεί status επιτυχίας από όλα τα URB του. Το USBD μπορεί να αναγνώσει το status κάθε IRP και να πραγματοποιήσει τις απαραίτητες ενέργειες.

5.4 Επικοινωνία με το Υλικό

Σε αυτή την παράγραφο παρουσιάζεται ο τρόπος με τον οποίο υλοποιήθηκαν οι διάφορες συναρτήσεις για την επίτευξη της επικοινωνίας μεταξύ του Host Controller Driver και ενός Host Controller, ο οποίος είναι προσαρτημένος στο υπολογιστικό σύστημα μέσω του PCI Bus.

Το πρώτο βήμα για την επικοινωνία μεταξύ του λογισμικού και υλικού ενός Συστήματος USB είναι το σάρωμα των συσκευών που είναι προσαρτημένες στον δίαυλο PCI (*PCI Bus*) και η ανίχνευση των συσκευών που αντιστοιχούν σε UHCI Host Controllers. Με την χρήση των διακοπών που παρουσάζονται στο Παράρτημα Γ μπορεί να διαβαστεί το PCI Configuration Space όλων των Functions που είναι συνδεδεμένες στο PCI Bus του υπολογιστικού συστήματος και να διαπιστωθεί εάν κάποια από αυτές υλοποιεί έναν UHCI Host Controller. Αρκεί να χρησιμοποιηθεί η διακοπή 1AB109 με Offset 00h για την λήψη του Vendor-ID του Function και η σύγκριση με την τιμή 1106h που είναι η ταυτότητα της Intel, και η ίδια διακοπή με Offset 02h για την λήψη του Device-ID και η σύγκριση με την τιμή 3038h που είναι η ταυτότητα του UHCI Host Controller.

Η διακοπή 1AB10A χρησιμοποιείται για την λήψη της διεύθυνης βάσης των καταχωρητών του UHCI Host Controller στο I/O Space του υπολογιστικού συστήματος χρησιμοποιώντας σαν Offset την τιμή 20h και άρα διαβάζοντας τον μεγέθους DWORD καταχωρητή USBBASE του UHCI Host Controller.

Τέλος, το λογισμικό υλοποιεί συναρτήσεις που χρησιμοποιούν τη διεύθυνση βάσης των καταχωρητών του Host Controller και το Offset κάθε καταχωρητή (βλέπε Ενότητα 4.2) για την εγγραφή δεδομένων σε αυτούς ή την ανάγνωση του περιεχομένου τους.

5.5 Συμπληρωματικό Λογισμικό για τον Έλεγχο της Ορθότητας της Υλοποίησης

Όπως έχει ήδη αναφερθεί, ο έλεγχος της ορθότητας του λογισμικού πραγματοποιήθηκε με την ανάπτυξη συμπληρωματικού λογισμικού που προσομοιώνει τη λειτουργία ενός UHCI Host Controller, σε ό,τι αφορά την επιστροφή του status των transactions του Schedule με το οποίο τροφοδοτείται το υλικό.

Το συμπληρωματικό λογισμικό θέτει πειραματικά Pipes και IRPs για τα Pipes αυτά για την τροφοδότηση του HCD με αιτήσεις μεταφοράς δεδομένων, ενώ επιστρέφει στις συναρτήσεις χρονοδρομολόγησης ένα ψευδοτυχαίο status για κάθε transaction του πειραματικού Schedule. Υποτίθεται ότι κατά τη λειτουργία του εικονικού Host Controller δεν προκύπτουν κρίσιμα σφάλματα που θα είχαν ως αποτέλεσμα την άμεση διακοπή της εκτέλεσης του Schedule (πχ Host Controller Process Error, Host System Error κτλ). Επίσης, υποτίθεται ότι ο εικονικός Host Controller δεν δέχεται σήματα που θα είχαν ως αποτέλεσμα την παύση της κανονικής λειτουργίας του (πχ EGSM, GRESET, HCRESET). Τέλος το συμπληρωματικό λογισμικό τυπώνει σε αρχεία κειμένου τα απαραίτητα μηνύματα για την επαλήθευση της ορθότητας των επεξεργασιών που εκτελεί το HCD.

Η παρούσα ενότητα περιορίζεται στην περιγραφή του αλγορίθμου επιστροφής του ψευδοτυχαίου status των transactions του Schedule. Ο αλγόριθμος χρονοδρομολόγησης του εικονικού Host Controller παρουσιάστηκε στην Παράγραφο 5.3.1, ενώ τα αρχεία κειμένου που τυπώνονται για τον έλεγχο της ορθότητας του HCD περιγράφονται στο Παράρτημα Δ.

5.5.1 Επιστροφή Ψευδοτυχαίου Status των Transactions

Σύμφωνα με τον αλγόριθμο του Σχήματος 5.3.1, το λογισμικό διατρέχει το Schedule κάθε frame που έχει χρονοδρομολογήσει για την λήψη του εικονικού status κάθε transaction, έτσι ώστε να ανασυντάξει τις λίστες των URB του εικονικού Host Controller και να χρονοδρομολογήσει το επόμενο frame του ίδιου Schedule Division στο οποίο ανήκει το frame το οποίο έχει εκτελεστεί. Βέβαια, η εκτέλεση του frame δεν είναι πραγματική αλλά εικονική: Το συμπληρωματικό λογισμικό δίνει ένα ψευδοτυχαίο status σε κάθε Transfer Descriptor του frame, χωρίς να πραγματοποιηθεί κάποια μεταφορά δεδομένων από κάποιον πραγματικό Host Controller.

Σε ό,τι αφορά τα Isochronous TDs, το συμπληρωματικό λογισμικό εκτελεί τις ίδιες επεξεργασίες με αυτές της περίπτωσης επιστροφής πραγματικού status: Το συμπληρωματικό λογισμικό διασχίζει όλα τα Isochronous TDs του frame και για κάθε ένα επιστρέφει στο αντίστοιχο IRP το status επιτυχίας (*return_iso_irp_status (EXECUTED)*). Αυτό γίνεται χρησιμοποιώντας τις πρόσθετες πληροφορίες του TD για να διασχίζει την λίστα των IRPs του Pipe, να βρει το συγκεκριμένο IRP, και να εγγράψει το status στο κατάλληλο πεδίο του IRP.

Σε ό,τι αφορά τα Non-Isochronous TDs, το λογισμικό διασχίζει τις ουρές του Non-Isochronous Schedule με όμοιο τρόπο όπως στον αλγόριθμο του Σχήματος 5.3.9. Το HCD διαβάζει τις πρόσθετες πληροφορίες του TD (πεδία λογισμικού) έτσι ώστε να βρει την ταυτότητα του Pipe και του IRP που αντιστοιχεί στο TD, και τον αριθμό του URB. Επίσης διαβάζει την τιμή του ορίου σφαλμάτων του TD και το μέγεθος των bytes που μεταφέρθηκαν στο transaction. Έπειτα, το λογισμικό θέτει ένα εικονικό status για το τρέχον TD με βάση την τιμή ενός ψευδοτυχαίου πραγματικού αριθμού (έστω random) που λαμβάνει τιμές στο διάστημα $0 \leq \text{random} \leq 1$. Η επιλογή του εικονικού status γίνεται με βάση τους ακόλουθους κανόνες, οι οποίοι τέθηκαν αυθαίρετα έτσι ώστε να είναι ρεαλιστικοί σε ό,τι αφορά την πιθανότητα να προκύψει η κάθε τιμή του status:

- Έαν ισχύει *0.99 < random*, τότε ανιχνέυθηκε σφάλμα Babble κατά την εκτέλεση του τρέχοντος TD.

- Αλλιώς, εάν ισχύει *0.97 < random*, τότε προκλήθηκε σφάλμα Stalled κατά την εκτέλεση του τρέχοντος TD.

- Αλλιώς, εάν ισχύει *0.95 < random*, τότε προκλήθηκε σφάλμα Bitstuff Error κατά την εκτέλεση του τρέχοντος TD.

- Αλλιώς, εάν ισχύει *0.90 < random*, τότε προκλήθηκε σφάλμα Buffer Error κατά την εκτέλεση του τρέχοντος TD.

- Αλλιώς, εάν ισχύει *0.80 < random*, τότε προκλήθηκε σφάλμα CRC/Timeout Error κατά την εκτέλεση του τρέχοντος TD.

- Αλλιώς, εάν ισχύει *0.75 < random*, τότε λήφθηκε χειραψία NAK από την συσκευή στόχο κατά την εκτέλεση του τρέχοντος TD.

- Αλλιώς, το TD εκτελέστηκε επιτυχώς.

Στο σημείο αυτό είναι απαραίτητο να σημειωθεί ότι όλα τα TDs σε κάθε frame προγραμματίζονται έτσι ώστε να εκτελεστούν από τον HC με τη μέθοδο διάσχισης Depth-First. Η αξία αυτής της παρατήρησης θα διαπιστωθεί παρακάτω. Το λογισμικό εκτέλει τις παρακάτω επεξεργασίες, οι οποίες διαφέρουν ελαφρώς από την περίπτωση της επιστροφής του status των transactions ενός πραγματικού Host Controller λόγω της εικονικής εκτέλεσης, ανάλογα με την τιμή του Status που επιλέγεται για το τρέχον TD:

- Εάν το status αντιστοιχεί σε σφάλμα Babble, το λογισμικό ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB αποσύρεται και επιστρέφει το αντίστοιχο status (*return_irp_status (ABORTED)*). Επίσης επιστρέφει σε όλα τα IRP του ίδιου Pipe status αποτυχίας λόγω σφάλματος σε άλλο IRP του ίδιου Pipe (*return_irp_status (ABORTED & FROM_OTHER)*). Το λογισμικό αφαιρεί από την λίστα των URBs όλα τα URB που αντιστοιχούν στο Pipe του αποτυχημένου URB. Τέλος, το λογισμικό επιστρέφει για τα επόμενα TDs του frame, που δεν ανήκουν στο Pipe στο οποίο αναφέρθηκε το σφάλμα, status επανεκτέλεσης (*return_irp_status (TO_BE_RETRIED & FROM_OTHER)*, θεωρώντας ότι δεν εκτελέστηκαν λόγω της μεθόδου διάσχισης Depth-First

(στην περίπτωση διάσχισης Breadth-First πιθανότητα θα υπάρχουν TDs σε επόμενες ουρές του Schedule τα οποία θα έχουν προλάβει να εκτελεστούν), καθώς ένας πραγματικός HC σταματάει την εκτέλεση των TDs του frame όταν ανιχνεύεται σφάλμα Babble σε κάποιο TD.

- Εάν το status αντιστοιχεί σε σφάλμα Stalled, το λογισμικό ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB αποσύρεται και επιστρέφει το αντίστοιχο status (*return_irp_status (ABORTED)*). Επίσης επιστρέφει σε όλα τα IRP του ίδιου Pipe status αποτυχίας λόγω σφάλματος σε άλλο IRP του ίδιου Pipe (*return_irp_status (ABORTED & FROM_OTHER)*). Το λογισμικό αφαιρεί από την λίστα των URBs όλα τα URB που αντιστοιχούν στο Pipe του αποτυχημένου URB. Το λογισμικό επιστρέφει για τα επόμενα TDs της ουράς στην οποία ανήκει το τρέχον TD, και τα οποία δεν ανήκουν στο Pipe στο οποίο αναφέρθηκε το σφάλμα, status επανεκτέλεσης (*return_irp_status (TO_BE_RETRIED & FROM_OTHER)*, θεωρώντας ότι δεν εκτελέστηκαν, κάτι που ισχύει σε έναν πραγματικό HC. Το λογισμικό συνεχίζει την επεξεργασία από την επόμενη ουρά.

- Εάν το status αντιστοιχεί σε σφάλμα BitStuff, CRC/Timeout, ή Buffer Error και το προηγούμενο όριο σφαλμάτων του τρέχοντος TD δεν είναι ίσο με 1, το λογισμικό ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB θα ξαναεκτελεστεί και επιστρέφει το αντίστοιχο status (return_irp_status (*TO_BE_RETRIED)*). Το αντίστοιχο URB παραμένει στην λίστα του με την νέα τιμή του ορίου σφάλματος (μειωμένη κατά ένα). Το λογισμικό επιστρέφει για τα επόμενα TDs της ουράς στην οποία ανήκει το τρέχον TD status επανεκτέλεσης (*return_irp_status (TO_BE_RETRIED & FROM_OTHER)*, θεωρώντας ότι δεν εκτελέστηκαν, κάτι που ισχύει σε έναν πραγματικό HC. Το λογισμικό συνεχίζει την επεξεργασία από την επόμενη ουρά.

- Εάν το status αντιστοιχεί σε σφάλμα BitStuff, CRC/Timeout, ή Buffer Error και το προηγούμενο όριο σφαλμάτων του τρέχοντος TD είναι ίσο με 1 (άρα το όριο σφαλμάτων θα ξεπεραστεί), το λογισμικό ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB αποσύρεται και επιστρέφει το αντίστοιχο status (*return_irp_status (ABORTED)*). Επίσης επιστρέφει σε όλα τα IRP του ίδιου Pipe status αποτυχίας λόγω σφάλματος σε άλλο IRP του ίδιου Pipe (*return_irp_status (ABORTED & FROM_OTHER)*). Το λογισμικό αφαιρεί από την λίστα των URBs όλα τα URB που αντιστοιχούν στο Pipe του αποτυχημένου URB. Το λογισμικό επιστρέφει για τα επόμενα TDs της ουράς στην οποία ανήκει το τρέχον TD, και τα οποία δεν ανήκουν στο Pipe στο οποίο αναφέρθηκε το σφάλμα, status επανεκτέλεσης (*return_irp_status (TO_BE_RETRIED & FROM_OTHER)*, θεωρώντας ότι δεν εκτελέστηκαν, κάτι που ισχύει σε έναν πραγματικό HC. Το λογισμικό συνεχίζει την επεξεργασία από την επόμενη ουρά.

- Εάν το status αντιστοιχεί σε λήψη χειραψίας NAK, το λογισμικό ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB θα ξαναεκτελεστεί και επιστρέφει το αντίστοιχο status (return_irp_status (*TO_BE_RETRIED)*). Το αντίστοιχο URB παραμένει στην λίστα του με το ίδιο όριο σφαλμάτων (Η λήψη χειραψίας NAK δεν θεωρείται κατάσταση σφάλματος). Το λογισμικό επιστρέφει για τα επόμενα TDs της ουράς στην οποία ανήκει το τρέχον TD

status επανεκτέλεσης (*return_irp_status (TO_BE_RETRIED & FROM_ OTHER)*, θεωρώντας ότι δεν εκτελέστηκαν, κάτι που ισχύει σε έναν πραγματικό HC. Το λογισμικό συνεχίζει την επεξεργασία από την επόμενη ουρά.

- Εάν το status αντιστοιχεί σε status επιτυχίας, το λογισμικό ενημερώνει το IRP στην λίστα του αντίστοιχου Pipe ότι το URB εκτελέστηκε επιτυχώς επιστρέφοντας τον αριθμό των bytes που μεταφέρθηκαν (*return_irp_status(EXECUTED)*). Το αντίστοιχο URB αφαιρείται από την λίστα του. Το λογισμικό συνεχίζει την επεξεργασία από το TD στο οποίο δείχνει το Link Pointer του τρέχοντος TD, ή από την επόμενη ουρά αν το Link Pointer δείχνει NULL.

ΚΕΦΑΛΑΙΟ 6

Συμπεράσματα και Μελλοντικές Κατευθύνσεις

Στο Κεφάλαιο αυτό επιχειρείται μια σύνοψη όλων όσων έχουν αναλυθεί στα πλαίσια της παρούσας Διπλωματικής Εργασίας και παρουσιάζονται τα σημαντικότερα συμπεράσματα που προκύπτουν από αυτή. Στη συνέχεια προτείνεται μια σειρά βημάτων για την ανάπτυξη ενός ολοκληρωμένου συστήματος λογισμικού για την υποστήριξη του Universal Serial Bus 1.1 σε ένα υπολογιστικό σύστημα υπό το λειτουργικό σύστημα MS-DOS.

6.1 Ανακεφαλαίωση και Συμπεράσματα

Στην παρούσα Διπλωματική Εργασία επιχειρήθηκε μια λεπτομερής ανάλυση των προδιαγραφών του Universal Serial Bus 1.1 σχετικά με το λογισμικό και το υλικό στην πλευρά του USB Host, με στόχο την ανάπτυξη ενός συστήματος λογισμικού, με βάση τις προδιαγραφές αυτές, για την κατάλληλη τροφοδότηση του υλικού του USB Host με τις αιτήσεις μεταφοράς δεδομένων προς εκτέλεση σε μορφή αναγνώσιμη από αυτό. Το λογισμικό αυτό συγγράφηκε για το λειτουργικό σύστημα MS-DOS, μπορεί όμως να χρησιμοποιηθεί με κάποιες τροποποιήσεις (που αφορούν την επικοινωνία με το υλικό) για παρόμοιες υλοποιήσεις σε άλλα λειτουργικά συστήματα.

Έτσι, αφού στο Κεφάλαιο 1 περιγράφονται οι εφαρμογές του USB 1.1 και κάποια βασικά χαρακτηριστικά της δομής του USB Host, στο Κεφάλαιο 2 περιγράφονται οι προδιαγραφές του Universal Serial Bus 1.1 σε ότι αφορά την δομή του USB Host, τον τρόπο ροής των δεδομένων μεταξύ των συνιστωσών του USB Host, οι τύποι μεταφοράς δεδομένων που υποστηρίζονται από το πρωτόκολλο και τα πεδία εφαρμογής τους, και η μορφή των συναλλαγών (*transactions*) στον δίαυλο USB για κάθε τύπο μεταφοράς δεδομένων.

Στο Κεφάλαιο 3 η ανάλυση επικεντρώθηκε ακόμα περισσότερο στον ρόλο και τον τρόπο επικοινωνίας μεταξύ των συνιστωσών (στρωμάτων) του USB Host, δίνοντας ιδιαίτερη έμφαση στην επικοινωνία μεταξύ του υλικού και του λογισμικού του USB Host, δηλαδή του Host Controller και του Host Controller Driver. Στη συνέχεια, επιχειρήθηκε μια γενική σύγκριση των ιδιοτήτων και των διαφορών των δύο διαθέσιμων υλοποιήσεων υλικού (Host Controller) του USB Host, του Universal Host Controller Interface και του Open Host Controller Interface.

Ακολούθως, στο Κεφάλαιο 4 παρουσιάστηκαν οι προδιαγραφές του Universal Host Controller Interface, της πιο διαδεδομένης τυποποίησης υλικού που χρησιμοποιείται για την υποστήριξη του USB 1.1, το οποίο για τον λόγο αυτό επιλέχθηκε ως το υλικό με βάση το οποίο σχεδιάστηκε το λογισμικό της εργασίας. Περιγράφτηκαν λεπτομερώς οι καταχωρητές μέσω των οποίων επιτυγχάνεται η επικοινωνία μεταξύ υλικού και λογισμικού του USB Host, οι δομές δεδομένων μέσω των οποίων περιγράφεται το χρονοδιάγραμμα (*Schedule*) των transactions που πρόκειται να εκτελεσούν από τον Host Controller, και ο τρόπος με τον οποίο το υλικό διασχίζει το Schedule για να εκτελέσει τις αιτήσεις μεταφοράς δεδομένων.

Με βάση τις προδιαγραφές αυτές που παρουσιάστηκαν στα Κεφάλαιο 2 και 4, αναπτύχθηκε ένα σύστημα λογισμικού για την οδήγηση ενός UHCI Host Controller δίνοντας έμφαση στα θέματα της υλοποίησης των απαραίτητων δομών δεδομένων για την αναπάσταση των αιτήσεων μεταφοράς δεδομένων (Pipes, IRPs και URBs), της

χρονοδρομόλογησης τους, και της παράδοσής τους στο υλικό, μετά τον μετασχηματισμό τους στη μορφή που απαιτείται από τις προδιαγραφές του UHCI. Οι λεπτομέρειες της υλοποίησης του λογισμικού αυτού παρουσιάζονται στο Κεφάλαιο 5.

Όπως έχει ήδη αναλυθεί λεπτομερώς στο Κεφάλαιο 5, ο λογισμικό που αναπτύχθηκε δεν επαρκεί για την οδήγηση ενός πραγματικού Host Controller, λόγω της μη υλοποίησης κάποιων σημαντικών συνιστωσών ενός πλήρους Host Controller Driver που έχουν να κάνουν με την διαχείριση του υλικού και την επικοινωνία μεταξύ υλικού και λογισμικού. Παρ' όλα αυτά, το λογισμικό αυτό υλοποιεί τις πιο ουσιώδεις δομές δεδομένων και επεξεργασίες ενός Host Controller Driver, οι οποίες αποτελούν και την καρδία του συστήματος USB. Αυτές είναι:

- Ο ορισμός των Pipes, τα οποία περιγράφουν τα χαρακτηριστικά των Endpoints τα οποία επικοινωνούν με τον USB Host, και κατ' επέκταση τα χαρακτηριστικά κάθε ροής δεδομένων μεταξύ του USB Host και των συνδεδεμένων συσκευών στον δίαυλο.

- Ο ορισμός των IRPs και των λιστών των IRPs κάθε Pipe, τα οποία αποτελούν το κύριο μέσο ροής επικοινωνίας μεταξύ των Clients, του USB Driver, και του Host Controller Driver, και αναπαριστούν τις αιτήσεις μεταφοράς δεδομένων στον δίαυλο.

- Η υλοποίηση ενός «δίκαιου» αλγορίθμου για την εξυπηρέτηση των εκκρεμών IRPs, δίνοντας σε κάθε Pipe την ίδια προτεραιότητα, κάτι το οποίο είναι βασική απαίτηση των προδιαγραφών του Universal Serial Bus 1.1.

- Ο ορισμός των URBs και ο τεμαχισμός των IRPs στα αντίστοιχα URBs με βάση τις προδιαγραφές του πρωτοκόλλου διαύλου του USB 1.1. Ο αναγνώστης μπορεί μελετώντας τους αλγορίθμους δημιουργίας των URBs να κατανοήσει τον τρόπο με τον οποίο ρέουν τα δεδομένα στον δίαυλο για κάθε τύπο μεταφοράς δεδομένων του USB 1.1. Επίσης, ο τεμαχισμός των IRPs σε URBs είναι απαραίτητη προϋπόθεση για την ορθή κατασκευή του Schedule ενός UHCI Host Controller.

- Ο υπολογισμός του αναμενόμενου χρόνου εκτέλεσης κάθε URB στον δίαυλο. Με βάση την εκτίμηση αυτή για κάθε URB πραγματοποείται η χρονοδρομολόγηση από το HCD.

- Η δέσμευση μνήμης για τις απαραίτητες δομές δεδομένων για κάθε Host Controller του συστήματος.

- Η υποστήριξη και των τεσσάρων τύπων μεταφοράς δεδομένων που ορίζονται από το USB 1.1: Isochronous, Interrupt, Control, και Bulk.

- Η δημιουργία ενός σκελετού ουρών του Schedule που επιτρέπει την τήρηση του πρωτοκόλλου Toggle Sequencing κάθε Endpoint, και επιπλέον διατηρεί μια ισορροπία μεταξύ του μήκους των ουρών και του αριθμού των ουρών από τις οποίες αποτελείται ο σκελετός. Υπενθυμίζεται ότι όσο πιο μεγάλο είναι το μήκος μιας ουράς, τόσο περισσότερα κατά μέσο όρο TD της ουράς δεν θα εκτελεστούν, λόγω αποτυχίας της εκτέλεσης κάποιου ανώτερου TD της ουράς.

- Η κατασκευή των Transfer Descriptors από τα αντίστοιχα URBs και η οδήγησή τους στο Schedule του Host Controller που πρόκειται να τα εκτελέσει, σύμφωνα με τις απαιτήσεις του UHCI.

- Η διασύνδεση των Transfer Descriptors και Queue Heads με σκοπό την δημιουργία του επιθυμητού Schedule.

- Η χρονοδρομολόγηση των URBs του Schedule, δεσμεύοντας τον χρόνο που προβλέπεται από τις προδιαγραφές του USB 1.1 για κάθε τύπο μεταφοράς δεδομένων.

- Η ανάγνωση του Status των transactions κάθε frame που έχει εκτελεστεί, με σκοπό την δυναμική χρονοδρομολόγηση των μελλοντικών frames και την επιστροφή του status στο IRP από το οποίο προέκυψε το transaction.

- Η παροχή των απαραίτητων συναρτήσεων για την αναγνώριση των Host Controllers στο PCI Bus του συστήματος, και την εγγραφή/ανάγνωση των καταχωρητών τους.

6.2 Μελλοντικές Κατευθύνσεις

Το λογισμικό που αναπτύχθηκε αποτελεί το κυριότερο μερός ενός UHCI Host Controller Driver για το λειτουργικό σύστημα MS-DOS. Αποτελεί όμως μια μόνο συνιστώσα του πλήρους συστήματος λογισμικού ενός USB Host. Επιπλέον το λογισμικό που αναπτύχθηκε εξαρτάται από την υλοποίηση του Host Controller που οδηγεί (UHCI). Παρακάτω προτείνεται μια σειρά από θέματα που θα πρέπει να μελετηθούν και να υλοποιηθούν, έτσι ώστε τελικά να έχει αναπτυχθεί ένα ολοκληρωμένο σύστημα λογισμικού για την υποστήριξη του Universal Serial Bus 1.1 στο λειτουργικό σύστημα MS-DOS:

- Ολοκλήρωση του UHCI Host Controller Driver με τη συγγραφή του λογισμικού οδήγησης του Root Hub και των συναρτήσεων διαχείρισης των διακοπών του UHCI Host Controller.

- Μελέτη του προτύπου Open Host Controller Interface και ανάπτυξη ενός OpenHCI Host Controller Driver για το λειτουργικό σύστημα MS-DOS.

- Ανάπτυξη ενός ολοκληρωμένου Universal Serial Bus Driver για το λειτουργικό σύστημα MS-DOS, και ενοποίηση των UHCI και OpenHCI HCDs και του USBD σε ένα USB System Software για το MS-DOS, το οποίο να διαχειρίζεται τα Pipes, τα Clients και τις λίστες των IRPs τους, το HCD κάθε Host Controller και τους buffers που ανατίθονται σε κάθε IRP, να ανιχνεύει το υλικό υποστήριξης του USB, και να διαχειρίζεται τις συσκευές που είναι προσαρτημένες στο USB.

- Μελέτη των προδιαγραφών του Universal Serial Bus 1.1 για τις συσκευές USB. Επιλογή συσκευών συμβατών με το πρωτόκολλο USB 1.1 για την υλοποίηση των Clients τους, που να προσαρμόζονται στο USB System Software που έχει αναπτυχθεί.

Μετά την ολοκλήρωση της ανάπτυξης του συστήματος λογισμικού για USB 1.1, επόμενο βήμα είναι η μελέτη των προδιαγραφών του Universal Serial Bus 2.0 και η ανάπτυξη ενός συστήματος λογιμικού για την υποστήριξη του USB 2.0 στο MS-DOS με βάση μια παρόμοια μεθοδολογία. Καθώς όμως η τεχνολογία USB είναι ιδιαίτερα δημοφιλής και συνεχώς αναπτυσσόμενη, στο μέλλον θα προκύψει η ανάγκη υλοποίησης λογισμικού υποστήριξης νέων, ακόμα πιο εξελιγμένων διαύλων της οικογένειας USB.

ΠΑΡΑΡΤΗΜΑ Α

Καταχωρητές του UHCI

Στο Παράρτημα Α παρουσιάζονται αναλυτικά οι καταχωρητές του UHCI, των οποίων η γενική λειτουργία περιγράφτηκε στην ενότητα 4.2. Περιγράφονται αναλυτικά τα bits των καταχωρητών και οι καταστάσεις που προκαλούν μεταβολή της τιμής τους, ή το αποτέλεσμα της ανάθεσης κάποιας τιμής σε ένα bit.

A.1 USB I/O Registers

A.1.1 USBCMD – USB Command Register

Πίνακας Α.1: USB Command Register

Bit	Description
15:8	**Reserved.**
7	**Max Packet.** 1=64 bytes. 0=32 bytes. Επιλέγει το μέγιστο μέγεθος πακέτου που επιτρέπεται να χρησιμοποιηθεί για reclamation του full speed bandwidth στο τέλος του frame. Αυτή η τιμή χρησιμοποιείται από τον HC για να καθορίσει εάν πρέπει να ξεκινήσει ένα νέο transaction βασιζόμενο στον χρόνο που απομένει στον SOF Counter. Η χρησιμοποίηση πακέτων προορισμένων για Bandwidth Reclamation μεγέθους μεγαλύτερου από αυτό που υποδεικνύει αυτό το bit θα προκαλέσει Babble Error αν εκτελεστεί κατά τη διάρκεια του Critical Window στο τέλος του frame, με αποτέλεσμα το σχετιζόμενο Endpoint να τεθεί σε κατάσταση Stalled. Το λογισμικό είναι υπεύθυνο για τη χρησιμοποίηση πακέτων μεγέθους μικρότερου από αυτό το όριο κατά τη φάση του Bandwidth Reclamation.
6	**Configure Flag (CF).** Το HCD θέτει αυτό το bit 1 σαν την τελευταία πράξη στην διαδικασία της ρύθμισης (Configuration) του HC. Το bit αυτό δεν επηρρεάζει το hardware. Χρησιμοποιείται μόνο ως σηματοφορέας για το software.
5	**Software Debug (SWDBG).** 1=Debug Mode, 0=Normal Mode. Στην SW Debug Mode ο HC θέτει 0 το Run/Stop bit μετά την ολοκλήρωση κάθε transaction. Το επόμενο transaction εκτελείται όταν το HCD θέσει 1 το Run/Stop bit. Το SWDBG bit πρέπει να γράφεται μόνο όταν ο HC βρίσκεται σε κατάσταση Stopped. Αυτό μπορεί να καθοριστεί ελέγχοντας το HCHalted bit στον καταχωρητή USBSTS.
4	**Force Global Resume (FGR).** 1=Ο HC στέλνει το σήμα Global Resume στο USB. Το HCD θέτει αυτό το bit 0 μετά από χρόνο 20 ms για να σταματήσει την αποστολή του σήματος. Σε αυτό το χρονικό διάστημα όλες οι USB συσκευές αναμένεται να είναι έτοιμες για αποστολή και λήψη δεδομένων στο δίαυλο. Ο HC θέτει 1 αυτό το bit όταν ανιχνεύσει ένα resume event (σύνδεση, αποσύνδεση κ.α.) ενώ βρίσκεται στο Global Suspend Mode. Η μετάβαση του bit αυτού από το 1 στο 0 (από το λογισμικό) προκαλεί την αποστολή από τις ports του HC ενός low speed EOP σήματος. Το bit θα παραμείνει 1 μέχρι να ολοκληρωθεί το σήμα EOP.
3	**Enter Global Suspend Mode (EGSM).** 1=Ο HC τίθεται σε Global suspend Mode. Σε αυτό το mode δεν πραγματοποιούνται transactions στο USB. Ο HC μπορεί να δέχεται σήματα από το USB και να στέλνει διακοπές στο σύστημα. Το λογισμικό θέτει αυτό το bit 0 για να θέσει τον HC εκτός Global Suspend Mode. Αυτό πρέπει να γίνεται ταυτόχρονα ή αφού τεθεί 0 το Force Global Resume bit. Επίσης το λογισμικό πρέπει να θέσει 0 το Run/Stop bit πριν θέσει 1 αυτό το bit.
2	**Global Reset (GRESET).** Όταν αυτό το bit είναι 1, ο HC στέλνει το σήμα Global

	Reset στο USB και κάνει reset όλους τους καταχωρητές του, συμπεριλαμβανομένων των εσωτερικών καταχωρητών των hubs. Το software θέτει 0 αυτό το bit μετά την πάροδο 10 ms τουλάχιστον, όπως ορίζεται από το USB Specification.
1	**Host Controller Reset (HCRESET).** Όταν αυτό το bit είναι 1, ο HC επαναφέρει τα εσωτερικά του timers, counters, state machines στις αρχικές του τιμές. Οποιοδήποτε transaction που βρίσκεται σε εξέλιξη τερματίζεται αμέσως. Ο HC επαναφέρει το bit αυτό στην τιμή 0 όταν ολοκληρωθεί η παραπάνω διαδικασία. Το HCReset επηρρεάζει τα bits [8,3:0] του καταχωρητή PORTSC κάθε port. Ο HC κάνει reset όλα τα state machines του, συμπεριλαμβανομένων των Connect/Disconnect state machines κάθε port, σηματοδοτώντας αποσύνδεση από τα ports, ακόμα και αν έχουν δυνδεδεμένη συσκευή. Αυτό έχει ως αποτέλεσμα το port να γίνει disabled και τα bits 1 και 3 του καταχωρητή PORTSC να γίνουν 1, και το bit 8 του ίδιου καταχωρητή να γίνει 0. Μετά από χρόνο 64 bits το HCReset bit γίνεται 0, ενεργοποιείται η δυνατότητα ανίχνευσης σύνδεσης και low-speed συσκευής και τα bits 0 και 8 του καταχωρητή PORTSC αλλάζουν ανάλογα.
0	**Run/Stop (RS).** 1=Run. 0=Stop.Όταν αυτό το bit είναι 1, ο HC εκτελεί το Schedule, όσο αυτό το bit παραμένει 1. Όταν το bit γίνει 0, ο HC ολοκληρώνει το transaction που βρίσκεται σε εξέλιξη και σταματά (halt). Τότε το HCHalted bit του καταχωρητή USBSTS γίνεται 1.Ο HC θέτει 0 αυτό το bit, όταν προκύψουν τα παρακάτω κρίσιμα (fatal) σφάλματα: Consistency Check Failure, PCI Bus Errors.

Ο παρακάτω πίνακας δίνει περεταίρω πληροφορίες για την λειτουργία των Run/Stop και Debug bits. Το SW Debug Mode παρέχει μια πολύ εύχρηστη μέθοδο για έλεγχο της ορθότητας του scheduling που πραγματοποιεί το HCD. Στο mode αυτό, τα ανενεργά *(inactive)* TD προσπερνούνται και ο HC γίνεται Halt μόνο αφού εκτελέσει ένα ενεργό *(active)* TD. Όταν εκτελεστεί το τελευταίο active TD του frame, ο HC περιμένει μέχρι να σταλεί το επόμενο SOF πακέτο και διαβάζει *(fetches)* το πρώτο active TD του επόμενου frame πριν καταστεί Halted.

Πίνακας Α.2: Run/Stop και Debug Bits

SWDBG	Run/Stop	Operation
0	0	Εάν ο HC εκτελεί κάποια εντολή, ολοκληρώνει την εντολή και μετά σταματά. Ο frame Counter γίνεται reset.
0	1	Η εκτέλεση επανέρχεται στην αρχή του frame, χρησιμοποιώντας τον frame list pointer που υποδεικνύεται από τον FRNUM Register. Η εκτέλεση του Schedule συνεχίζεται μέχρι το Run/Stop bit γίνει 0.
1	0	Εάν ο HC εκτελεί κάποια εντολή, ολοκληρώνει την εντολή και μετά σταματά. Ο frame Counter παγώνει στην τρέχουσα τιμή του.
1	1	Η εκτέλεση επανέρχεται στο σημείο που είχε τερματιστεί προηγουμένως (στο ίδιο σημείο του frame και του schedule). Το Run/Stop bit γίνεται 0 όταν ο HC διαβάσει ένα TD, και σταματά (halt) αφού εκτελεσει το TD, θέτοντας 1 το HCHalted bit.

Εκτός SW Debug mode το HCHalted bit χρησιμοποιείται για να υποδείξει πότε ο HC έχει ανιχνεύσει το μηδενικό Run/Stop bit και έχει ολοκληρώσει το τρέχον

transaction. Σε κανονικό mode, όποτε το Run/Stop bit τίθεται 0, ο SOF Counter μηδενίζεται με αποτέλεσμα η εκτέλεση να συνεχίζεται από την αρχή του frame, στο entry του Frame List που υποδεικνύει ο FRNUM Register,και όχι από εκεί που σταμάτησε.

A.1.2 USBSTS – USB Status Register

Πίνακας A.3: USB Status Register

Bit	Description
15:6	**Reserved.**
5	**HCHalted.** Ο HC θέτει 1 αυτό το bit αφού το Run/Stop bit τεθεί 0, είτε από το HCD, είτε από το υλικό (Debug Mode ή Internal Error)
4	**Host Controller Process Error.** Ο HC θέτει 1 αυτό το bit όταν ανιχνεύσει ένα κρίσιμο σφάλμα (Fatal Error) και δηλώνει ότι ο HC υπέστει Consistency Check Failure, κατά την επεξεργασία ενός TD (πχ μη επιτρεπτή τιμή στο πεδίο PID). Ο HC μηδενίζει το Run/Stop bit στον USBCMD και παράγεται μια διακοπή υλικού στο σύστημα.
3	**Host System Error.** Ο HC θέτει 1 αυτό το bit όταν ανιχνεύσει ένα σημαντικό σφάλμα στον Host που σχετίζεται με τον HC. Σε ένα σύστημα PCI, το bit αυτό γίνεται 1 εξαιτίας των εξής καταστάσεων: PCI Parity Error, PCI Master Abort και PCI Target Abort. Ο HC μηδενίζει το Run/Stop bit στον USBCMD και παράγεται μια διακοπή υλικού στο σύστημα.
2	**Resume Detect.** Ο HC θέτει 1 αυτό το bit όταν ανιχνεύσει ένα σήμα RESUME από μια συσκευή USB. Αυτό συμβαίνει μόνο στο Global Suspend Mode (bit 3 του USBCMD).
1	**USB Error Interrupt.** Ο HC θέτει 1 αυτό το bit όταν η ολοκλήρωση ενός transaction έχει ως αποτέλεσμα μια κατάσταση σφάλματος. Έαν το αντίστοιχο TD είχε το IOC bit του 1, τότε και το bit 0 του καταχωρητή τίθεται 1.
0	**USB Interrupt (USBINT).** Ο HC θέτει 1 αυτό το bit κατά την ολοκλήρωση ενός transaction που το αντίστοιχο TD είχε το IOC bit του 1 ή όταν ο HC ανίχνευσε ένα Short Packet σε ένα transaction που το αντίστοιχο TD είχε 1 το bit Short Packet Detect.

A.1.3 USBINTR – USB Interrupt Enable Register

Πίνακας A.4: USB Interrupt Enable Register

Bit	Description
15:4	**Reserved.**
3	**Short Packet Interrupt Enable.** 1=Enabled. 0=Disabled.
2	**Interrupt On Complete (IOC) Enable.** 1=Enabled. 0=Disabled.
1	**Resume Interrupt Enable**. 1=Enabled. 0=Disabled.
0	**Timeout/CRC Interrupt Enable.** 1=Enabled. 0=Disabled.

A.1.4 FRNUM – Frame Number Register

Ο καταχωρητής FRNUM δεν μπορεί να γραφτεί παρά μόνο εάν ο Host Controller βρίσκεται στην κατάσταση STOPPED (STOPPED State), όπως δηλώνεται από το HCHalted bit (USBSTS Register). Σε αντίθετη περίπτωση, δηλαδή εάν το Run/Stop bit είναι 1, οι εγγραφές στον καταχωρητή αγνοούνται.

Πίνακας A.5: Frame Number Register

Bit	Description
15:11	**Reserved.**
10:0	**Frame List Current Index/Frame Number.** Τα bits [10:0] παρέχουν το frame number του πακέτου SOF του frame. Η τιμή του καταχωρητή αυξάνεται στο τέλος του χρόνου κάθε frame (περίπου κάθε 1 ms). Επιπλέον τα bits [9:0] χρησιμοποιούνται ως τρέχων δείκτης στο Frame List και αντιστοιχούν στα σήματα διεύθυνσης μνήμης (memory address signals) [11:2].

A.1.5 FRBASEADD – Frame List Base Address Register

Πίνακας A.6: Frame List Base Address Register

Bit	Description
31:12	**Base address.** Τα bits αυτά αντιστοιχούν στα σήματα διεύθυνσης μνήμης (memory address signals) [31:12].
11:0	**Reserved.** Πρέπει να γράφονται ως 0.

A.1.6 SOFMOD – Start of Frame Modify Register

Πίνακας A.7: Start Of Frame Modify Register

Bit	Description
7	**Reserved.**
6:0	**SOF Timing Value.** Η συχνότητα παραγωγής των SOF πακέτων (SOF Cycle Time) ισούται με (11936 + τιμή του πεδίου) KHz. Η default τιμή είναι 64 (δεκαδική) η οποία δίνει SOF Cycle time ίσο με 12 MHZ στην είσοδο του ρολογιού, με αποτέλεσμα την παραγωγή Frames με διάρκεια 1 ms. Ο παρακάτω πίνακας δίνει την αντιστοιχία μεταξύ του SOF Cycle Time και της τιμής του καταχωρητή. **Frame Length** **SOFMOD Value** (δεκαδικές τιμές) 11936 0 11937 1 . . . . 11999 63 12000 64

	12001	65
	.	.
	.	.
	12062	126
	12063	127

Σημειώνεται ότι καταχωρητής SOFMOD γίνεται reset όταν γίνει Host Controller reset ή Global Reset.

A.1.7 PORTSC – Port Status and Control Register

Μετά από ένα Power-Up Reset, Global Reset ή HC Reset, οι αρχικές καταστάσεις σε ένα Port είναι: Μη συνδεδεμένη συσκευή, Port Disabled, και το Bus Line Status είναι 00. Αν μια συσκευή είναι συνδεδεμένη στο port, η κατάσταση θα μεταβεί στην κατάσταση σύνδεσης και το λογισμικό του συστήματος θα επεξεργαστεί τη μετάβαση αυτή όπως οποιαδήποτε ένδειξη αλλαγής status. Η μετάβαση αυτή συμβαίνει μετά από χρόνο τουλάχιστον 64 bits. Αν ο HC βρίσκεται σε Global Suspend Mode, τότε αν ένα οποιοδήποτε bit από τα [6,3,1] τεθεί 1, τότε ο HC θα στείλει ένα σήμα Global Resume.

Πίνακας Α.8: Port Status and Control Register

Bit	Description
15:13	**Reserved.** Πρέπει να γράφονται ως 0.
12	**Suspend – R/W.** 1=To Port είναι σε κατάσταση Suspend. 0=To Port δεν είναι σε κατάσταση Suspend. Το bit αυτό είναι 1 εάν ο HC βρίσκεται σε Global Suspend Mode. Σε συνδυασμό με το bit 2, η κατάσταση ενός hub μπορεί να είναι: **Bits [12,2]** **Hub State** x0 Disable 01 Enable 11 Suspend Στο Suspend Mode, η ροή δεδομένων από τον HC προς το USB μπλοκάρεται σε αυτό το Port, εκτός από τα σήματα Global Reset και Port Reset. Πριν την είσοδο στο Suspend Mode, ολοκληρώνεται το τρέχον transaction και μετά αλλάζει η τιμή του bit.
11:10	**Reserved.**
9	**Port Reset – R/W.** 1=H Port είναι σε κατάσταση Reset. 0=H Port δεν είναι σε κατάσταση Reset. Όταν το port είναι σε κατάσταση Reset, η port είναι Disabled και στέλνει το σήμα USB Reset. Το λογισμικό πρέπει να εγγυηθεί ότι το σήμα αυτό είναι ενεργό για το κατάλληλο χρονικό διάστημα, όπως ορίζει το USB Specification.
8	**Low Speed Device Attached – RO.** 1=Low Speed συσκευή είναι συνδεδεμένη στο Port. 1=Full Speed συσκευή. Οι εγγραφές στο bit αυτό αγνοούνται.
7	**Reserved-RO.** Διαβάζονται ως 1.
6	**Resume Detect – R/W.** 1=Ανιχνεύθηκε/οδηγήθηκε Resume στο Port. 0=Δεν ανιχνεύθηκε/οδηγήθηκε Resume στο Port. Η εγγραφή ενός 0, από 1, έχει ως

	αποτέλεσμα το Port να στείλει ένα Low Speed EOP. Το bit θα παραμείνει 1 μέχρι να ολοκληρωθεί η αποστολή του EOP.
5:4	**Line Status – RO.** Τα bits αυτά χρησιμοποιούνται για ανίχνευση λαθών και ανάκυψη από λάθη και για διαγνωστικές λειτουργίες του USB. Το πεδίο αυτό ενημερώνεται στο χρονικό σημείο EOF2.
3	**Port Enable/Disable Change – R/WC.** 1=Το Port enabled/disabled status άλλαξε. 0=Καμία αλλαγή. Για το Root Hub, αυτό το bit γίνεται 1 μόνο όταν το Port γίνεται Disabled λόγω αποσύνδεσης ή λόγω ειδικών συνθηκών στο σημείο EOF2.
2	**Port Enabled/Disabled – R/W.** 1=Enable. 0=Disable. Τα Ports γίνονται Enabled μόνο από το λογισμικό. Μπορούν να γίνουν Disabled είτε από μια κατάσταση σφάλματος ή από το λογισμικό. Το status αυτού του bit αλλάζει μετά την ολοκλήρωση του τρέχοντος transaction, αν υπάρχει.
1	**Connect Status Change – R/WC.** 1=Αλλαγή στο παρόν Connect Status. 0=Καμία Αλλαγή.
0	**Current Connect Status – RO.** 1=Συσκευή παρούσα στο Port. 0=Δεν υπάρχει συνδεδεμένη συσκευή. Το bit αυτό παίρνει την κατάλληλη τιμή μόλις συμβεί το γεγονός που αλλάζει το Connect Status, θέτοντας 1 το bit 1.

Όπως γίνεται εύκολα αντιληπτό, οι καταχωρητές PORTSC δίνουν την τρέχουσα κατάσταση ενός Port, και όχι απαραίτητα την τιμή που γράφτηκε τελευταία στους καταχωρητές. Αυτή η ιδιότητα επιτρέπει στο HCD να κάνει polling στους καταχωρητές αυτούς και να περιμένει την κατάλληλη κατάσταση του Port για να κάνει τις αντίστοιχες ενέργειες.

A.2 PCI Configuration Registers

A.2.1 CLASSC – Class Code Register

Εάν το πεδία Base Class Code και Sub-Class Code του καταχωρητή CLASSC είναι 0Ch και 03h αντίστοιχα, τότε η συσκευή υλοποιεί ένα USB Host Controller.

Πίνακας Α.9: Class Code Register

Bit	Description
23:16	**Base Class Code (BASEC).** 0Ch=Serial Bus Controller
15:8	**Sub-Class Code (SCC).** 03=Universal Serial Bus Host Controller
7:0	**Programming Interface (PI).** 00h=Δεν ορίζεται συγκεκριμένο programming interface σε επίπεδο καταχωρητών.

A.2.2 USBBASE – I/O Space Base Address Register

Πίνακας Α.10: I/O Space Base Address Register

Bit	Description
31:16	**Reserved.** Καλωδιωμένα στο 0. Πρέπει να γράφονται ως 0.
15:5	**Index Register Base Address.** Τα bits [15:5] του καταχωρητή αντιστοιχούν στα σήματα διεύθυνσης (address signals AD) [15:5] και δίνουν τη διεύθυνση βάσης των καταχωρητών USB I/O της συσκευής . Τα σήματα [4:0] θεωρούνται ίσα με 0.
4:1	**Reserved.** Διαβάζονται ως 0.
0	**Resource Type Indicator (RTE)-RO.** Καλωδιωμένο στο 1 δηλώνοντας ότι το πεδίο διεύθυνσης βάσης του καταχωρητή δείχνει στο I/O Space.

A.2.3 SBRN - Serial Bus Release Number Register

Πίνακας Α.11: Serial Bus Release Number Register

Bit	Description
7:0	**Serial Bus Specification Number.** Οι διάφοροι συνδυασμοί bits αντιστοιχούν σε διαφορετικές εκδόσεις του πρωτοκόλλου. **Bits [7:0]** **Release Number** 00h Pre-release 1.0 10h Release 1.0

ΠΑΡΑΡΤΗΜΑ Β

Δομές Δεδομένων του UHCI

Στο Παράρτημα Β παρουσιάζονται οι δομές δεδομένων του UHCI, των οποίων τα γενικά χαρακτηριστικά παρουσιάστηκαν στην ενότητα 4.3. Περιγράφονται αναλυτικά τα bits των δομών δεδομένων και οι καταστάσεις που προκαλούν μεταβολή της τιμής τους, ή το αποτέλεσμα της ανάθεσης κάποιας τιμής σε ένα bit.

B.1 Frame List Pointer

Πίνακας B.1: Frame List Pointer

Bit	Description
31:4	**Frame List Pointer (FLP).** Αυτό το πεδίο περιέχει τη διεύθυνση του πρώτου Data Object για επεξεργασία στο Frame και αντιστοιχεί στα σήματα διεύθυνσης (memory signals) [31:4]. Τα σήματα [3:0] θεωρούνται μηδενικά.
3:2	**Reserved.** Πρέπει να γράφονται ως 0.
1	**QH/TD Select.** 1=QH. 0=TD. Αυτό το bit δηλώνει στο hardware αν το Data Object στο οποίο αναφέρεται το FLP είναι QH ή TD, ώστε να το επεξεργαστεί κατάλληλα αφού το διαβάσει.
0	**Terminate (T).** 1=Κενό frame (Invalid Pointer). 0=Δείχνει σε ένα QH ή TD (Valid Pointer). Αυτό το bit δείχνει στο hardware αν το Schedule σε αυτό το frame έχει έγκυρα Data Objects.

B.2 Transfer Descriptor

B.2.1 TD Link Pointer

Πίνακας B.2: TD Link Pointer

Bit	Description
31:4	**Link Pointer (LP).** Τα bits αυτά αντιστοιχούν στα σήματα διεύθυνσης (memory signals) [31:4]. Αυτό το πεδίο δείχνει το επόμενο TD ή QH.
3	**Reserved.** Πρέπει να γράφονται ως 0.
2	**Depth/Breadth Select (Vf).** 1=Depth First. 0=Breadth First. Αυτό το bit είναι έγκυρο μόνο για TD που βρίσκονται στην ουρά ενός QH. Υποδεικνύει αν η εκτέλεση των TD πρέπει να συνεχιστεί στην ίδια ουρά (Depth First) ή να συνεχίσει από την ουρά του επόμενου QH (Breadth First).
1	**QH/TD Select (Q).** 1=QH. 0=TD. Αυτό το bit δηλώνει στο hardware αν το Data Object στο οποίο αναφέρεται το LP είναι QH ή TD, ώστε να το επεξεργαστεί κατάλληλα αφού το διαβάσει.
0	**Terminate (T).** 1=Invalid Link Pointer. 0=Valid Link Pointer. Ενημερώνει τον HC αν το LP δείχνει σε έγκυρο Data Object ή όχι. Αν το TD βρίσκεται στην ουρά ενός QH σημαίνει ότι αυτό είναι το τελέυταίο TD της ουράς. Αν βρίσκεται εκτός ουράς σημαίνει ότι αυτό είναι το τελευταίο TD του Frame.

B.2.2 TD Control and Status

Πίνακας Β.3: TD Control and Status

Bit	Description
31:30	**Reserved (R).**
29	**Short Packet Detect (SPD).** 1=Enable. 0=Disable. Όταν ένα πακέτο έχει αυτό το bit 1, έχει κατεύθυνση προς τον Host (input packet), το TD του είναι σε ουρά, και ολοκληρώνεται επιτυχώς με actual length μικρότερο από το maximum length, τότε το TD γίνεται inactive, το Queue Header δεν ενημερώνεται και το USBINT bit του USBSTS γίνεται 1 στο τέλος του frame, στέλνοντας την αντίστοιχη διακοπή εάν είναι enabled. Αυτό προϋποθέτει ότι δεν θα συμβεί κάποιο σφάλμα. Εάν είναι output packet ή εκτός ουράς, η συμπεριφορά είναι αόριστη.
28:27	**C_Err.** Αυτό το πεδίο μετράει τον αριθμό των σφαλμάτων που προκύπτουν κατά τη διάρκεια του transaction. Τα σφάλματα μετρώνται αντίστροφα. Εάν η αρχική τιμή του πεδίου είναι μη μηδενική και ο μετρητής γίνεται 0, το TD γίνεται inactive και ο HC θέτει το Stalled bit 1, καθώς και το bit που αντιστοιχεί στο error που προκλήθηκε. Επίσης θα σταλεί το αντίστοιχο interrupt (αν είναι enabled στον καταχωρητή USBINTR). Εάν η αρχική τιμή του πεδίου είναι 0, τότε ο HC δε θα μετράει τα σφάλματα και το TD θα επαναλαμβάνεται μέχρι να ολοκληρωθεί επιτυχώς. Το πεδίο αυτό έχει συνήθως αρχική τιμή 11 για non-isochronous TD. **Bits [28:27]** — **Interrupt After** 00 — No Error Limit 01 — 1 Error 10 — 2 Errors 11 — 3 Errors **Error** — **Decrement Counter** CRC Error — Yes Timeout Error — Yes NAK Received — No Babble Detected — No, TD inactive Data Buffer Error — Yes Stalled — No, TD inactive Bitstuff Error — Yes
26	**Low Speed Device (LS).** 1=Η συσκευή στόχος είναι Low Speed συσκευή. 0=Full Speed συσκευή. Εάν σε ένα port του root hub είναι συνδεδεμένη μια full speed συσκευή και αυτό το bit είναι 1, τότε ο HC θα στείλει ένα low speed preamble στο αντίστοιχο port προτού στείλει το PID.
25	**Isochronous Select (IOS).** 1=Isochronous TD. 0=Non-Isochronous TD. Δηλώνει αν το transfer type είναι isochronous ή όχι.
24	**Interrupt on Complete (IOC).** 1=Στείλε IOC. Όταν αυτό το bit είναι, ο HC θα στείλει μια διακοπή στο τέλος του frame ακομά και αν το TD είναι inactive και δεν υπάρξει transaction.
23:16	**Status.** Τα bit αυτού του πεδίου χρησιμοποιούνται για αναφορά από τον HC του status του transaction που αντιστοιχεί στο TD. Για isochronous TD το status είναι πάντα Inactive. Για τα υπόλοιπα TD, για τα οποία γίνεται έλεγχος λαθών και επαναλήψεις, το status έχει μεγάλη σημασία. **Bit** — **Status Field Description** 23 — **Active.** Πριν από την εκτέλεση του TD, το HCD θέτει 1 αυτό το bit. Ο HC θέτει 0 αυτό το bit εάν το transaction ολοκληρωθεί, για να μην εκτελεστεί ξανά. Επίσης τίθεται 0 από τον HC να ληφθεί μια

	χειραψία STALL από το endpoint στόχο. 22 **Stalled.** Τίθεται 1 εάν στο αντίστοιχο transaction αναφερθεί babble error ή ληφθεί μια χειραψία STALL από τη συσκευή. Εάν η STALL λαμβάνεται σε ένα setup transaction θα αναφερθεί και ένα Time Out Error. 21 **Data Buffer Error.** Τίθεται 1 εάν στον buffer που σχετίζεται με το transaction συμβεί underrun (κατά την αποστολή δεδομένων) ή overrun (κατά τη λήψη δεδομένων). Σε κάθε περίπτωση το actual length δεν θα είναι ίσο με το maximum length. Σε περίπτωση ίπτωση run ο HC θα στείλει λάθος CRC και θα αφήσει το TD active. Σε περίπτωση overrun θα συμβεί ένα timeout error. 20 **Babble Detected.** Τίθεται 1 όταν συμβεί Babble Error. Τότε θα τεθεί 1 και το Stalled bit. Σταματάει αμέσως η εκτέλεση των TD του Frame. Η εκτέλεση συνεχίζεται από το επόμενο frame. 19 **NAK Received.** Τίθεται 1 όταν ληφθεί πακέτο NAK κατά το transaction. Αν το NAK ληφθεί σε ένα SETUP transaction θα αναφερθεί και Time Out Error. 18 **CRC/Time Out Error.** Τίθεται 1 από τον HC όταν συμβεί CRC Error ή όταν η συσκευή δεν απαντήσει στο Status Update μέσα στο χρονικό διάστημα που ορίζεται από το USB Specification. Άλλες καταστάσεις που περιγράφονται σε άλλα bits μπορούν να θέσουν 1 αυτό το bit. 17 **Bitstuff Error.** Τίθεται 1 από τον HC όταν το λαμβανόμενο bit stream έχει 6 συνεχόμενα 1. 16 **Reserved (R).**
15:11	**Reserved (R).**
10:0	**Actual Length (ActLen).** Το πεδίο αυτό γράφεται από τον HC κατά την ολοκλήρωση του transaction, για να δηλώσει τον αριθμό των bytes που μεταφέρθηκαν. Το λογισμικό χρησιμοποιεί το πεδίο αυτό για να ελέγξει την ακεραιότητα των δεδομένων. Το πεδίο κωδικοποιείται στη μορφή n-1.

B.2.3 TD Token

Πίνακας B.4: TD Token

Bit	Description
31:21	**Maximum Length (MaxLen).** Καθορίζει το μέγιστο αριθμό bytes δεδομένων (όχι bytes πρωτοκόλλου) που επιτρέπονται για τη μεταφορά. Ο θεωρητικά μέγιστος αριθμός bytes δεδομένων είναι 1280 bytes, όμως το USB Specification ορίζει το μέγιστο αριθμό bytes ίσο με 1023 bytes. Η κωδικοποίηση του πεδίου γίνεται στη μορφή n-1. Η τιμή 7FFh δίνει 0 bytes. Οι επιτρεπόμενες τιμές είναι 00h-4FFh και η τιμή 7FFh. Τιμές πέρα από τις επιτρεπόμενες προκαλούν consistency check failure. Σε περίπτωση αποστολής δεδομένων από τον HC, το πεδίο αυτό αντιπροσωπεύει τον αριθμό των bytes που διαβάστηκαν από τη μνήμη. Στις περισσότερες περιπτώσεις αυτός είναι ο αριθμός των bytes που θα μεταδοθούν. Σε κάποιες σπάνιες περιπτώσεις ο HC μπορεί να μην έχει πρόσβαση στη μνήμη στον

	κατάλληλο χρόνο, για να μην γίνει underrun. Σε αυτή την περίπτωση ο HC θα στείλει λιγότερα bytes από αυτά που δηλώνονται σε αυτό το πεδίο. Σε πείπτωση λήψης δεδομένων από τον HC, το πεδίο αυτό αντιπροσωπεύει τον μέγιστο αριθμό bytes που θα έπρεπε να στείλει η USB συσκεύη στον HC. Αν η συσκευή συνεχίσει να στέλνει δεδομένα, αφού ο HC έχει λάβει MaxLen bytes, θα δημιουργηθεί BABBLE error.
20	**Reserved (R).**
19	**Data Toggle (D).** 0=DATA0, 1=DATA1. Περιέχει το Data PID που θα σταλεί ή αναμένεται. Για isochronous TD πρέπει να είναι πάντα 0.
18:15	**Endpoint (EndPt).** Περίεχει τον αριθμό του Endpoint της συσκευής που σχετίζεται με το transaction, το οποίο θα στείλει ή θα λάβει δεδομένα.
14:8	**Device Address.** Αυτό το πεδίο περιέχει την USB διεύθυνση της συσκευής που σχετίζεται με το transaction.
7:0	**Packet Identification (PID).** Αυτό το πεδίο περιέχει το Packet ID που θα χρησιμοποιηθεί στο transaction. Επιτρέπονται μόνο οι τιμές IN (69h), OUT (E1h), και SETUP (2Dh). Διαφορετικά προκαλείται Consistency Check Failure, και ο HC θα γίνει αμέσως Halt. Τα bits [3:0] είναι συμπληρώματα των bits [7:4]. Σημειώνεται ότι πακέτα διαφορετικού PID από IN, OUT ή SETUP *(Handshakes)* στέλνονται αυτόματα από τον HC.

B.2.4 TD Buffer Pointer

Πίνακας B.5: TD Buffer Pointer

Bit	Description
31:0	**Buffer Pointer.** Τα bits [31:0] αντιστοιχούν στα σήματα διεύθυνσης (memory signals) [31:0].

B.3 Queue Head

B.3.1 Queue Head Link Pointer

Πίνακας B.6: Queue Head Link Pointer

Bit	Description
31:4	**Queue Head Link Pointer (QHLP).** Αυτό το πεδίο περιέχει τη διεύθυνση του επόμενου Data Object προς επεξεργασία στην οριζόντια λίστα και αντιστοιχεί στα σήματα διεύθυνσης (memory signals) [31:4].
3:2	**Reserved.** Πρέπει να γράφονται ως 0.
1	**QH/TD Select (Q).** 1=QH, 0=TD. Αυτό το bit δηλώνει στο hardware αν το Data Object στο οποίο αναφέρεται το QHLP είναι QH ή TD, ώστε να το επεξεργαστεί κατάλληλα αφού το διαβάσει.
0	**Terminate (T).** 1=Invalid Link Pointer. 0=Valid Link Pointer. Ενημερώνει τον HC αν το QH είναι το τελευταίο στο Schedule. Αν υπάρχουν active TD στην ουρά, θα είναι τα τελευταία που θα εκτελεστούν στο Frame.

B.3.2 Queue Element Link Pointer

Πίνακας B.7: Queue Element Link Pointer

Bit	Description
31:4	**Queue Element Link Pointer (QELP).** Αυτό το πεδίο περιέχει τη διεύθυνση του πρώτου Data Object προς επεξεργασία στην ουρά και αντιστοιχεί στα σήματα διεύθυνσης (memory signals) [31:4].
3	**Reserved.** Πρέπει να γράφεται ως 0.
2	**Reserved.** Δεν επιτελεί καμία λειτουργία. Η τιμή του μπορεί να αλλάζει σαν παρενέργεια του update του Queue Element Link Pointer.
1	**QH/TD Select (Q).** 1=QH, 0=TD. Αυτό το bit δηλώνει στο hardware αν το Data Object στο οποίο αναφέρεται το QELP είναι QH ή TD, ώστε να το επεξεργαστεί κατάλληλα αφού το διαβάσει.
0	**Terminate (T).** 1=Terminate (καμία valid entry στην ουρά). Αυτό το bit ενημερώνει τον HC ότι δεν υπάρχουν έγκυρα TD στην ουρά.

ΠΑΡΑΡΤΗΜΑ Γ

Λήψη Πληροφοριών από το PCI Configuration Space στο Λειτουργικό Σύστημα MS-DOS

Οι συσκευές που συνδεόνται σε ένα υπολογιστικό σύστημα μέσω του PCI Bus διαθέτουν ένα σύνολο καταχωρητών για την λήψη των απαραίτητων πληροφοριών από το λογισμικό του συστήματος για την αναγνώριση και ρύθμισή τους. Το σύνολο των καταχωρητών αυτών αναφέρεται ως PCI Configuration Space. Το περιεχόμενο κάθε καταχωρητή του Configuration Space έχει μια μοναδική σημασία που ορίζεται από το πρότυπο PCI. Η λεπτομερής περογραφή των καταχωρητών του PCI Configuration Space είναι εκτός θέματος του παρόντος συγγράμματος. Η ανάλυση που ακολουθεί περιορίζεται στα σημεία που ενδιαφέρουν.

Στο λειτουργικό σύστημα MS-DOS υλοποιούνται δύο πολύ χρήσιμες Διακοπές Λογισμικού (*Software Interrupts*) για την λήψη πληροφοριών από το Configuration Space μιας Λειτουργίας (*Function*) που υλοποιεί μια συσκευή που είναι προσαρτημένη στο υπολογιστικό σύστημα μέσω του PCI Bus. Οι διακοπές αυτές περιγράφονται στον παρακάτω πίνακα:

Πίνακας Γ.1: Διακοπές Λογισμικού για την Ανάγνωση του PCI Configuration Space

Διακοπή	Τιμές Καταχωρητών	Επιστροφή
1AB109	AX = BI09h	CX= Περιεχόμενο Καταχωρητή που αναγνώστηκε (WORD)
	BH = Αριθμός του Bus (Bus Number)	
	BL = Αριθμός Device/Function, Bits 7-3: Device, Bits 2-0: Function	
	DI = Offset του Καταχωρητή προς Ανάγνωση (μεγέθους WORD)	
1AB10A	AX = BI0Ah	ECX= Περιεχόμενο Καταχωρητή που αναγνώστηκε (DWORD)
	BH = Αριθμός του Bus (Bus Number)	
	BL = Αριθμός Device/Function, Bits 7-3: Device, Bits 2-0: Function	
	DI = Offset του Καταχωρητή προς Ανάγνωση (μεγέθους DWORD)	

Σημειώνεται ότι στο Offset 00h του PCI Configuration Space ενός Function περιέχεται το Vendor-ID (*Ταυτότητα Κατασκευαστή*), μεγέθους WORD, του Function. Επίσης, στο Offset 02h του PCI Configuration Space ενός Function περιέχεται το Device-ID (*Ταυτότητα Συσκευής*), μεγέθους WORD, του Function. Στα offsets αυτά αποθηκεύονται όλες οι απαραίτητες πληροφορίες για τον έλεγχο εάν το υπό εξέταση Function αποτελεί έναν UHCI Host Controller. Για να ισχύει το παραπάνω αρκεί: Vendor-Id= 1106h και Device-ID= 3038.

ΠΑΡΑΡΤΗΜΑ Δ

Συνοδευτικό CD-ROM της Διπλωματικής Εργασίας

Στο συνοδευτικό CD-ROM της Διπλωματικής Εργασίας περιέχονται τα αρχεία του πηγαίου κώδικα στην γλώσσα προγραμματισμού C, το αντίστοιχο εκτελέσιμο αρχείο, τα αρχεία που περιέχουν τα αποτελέσματα της εκτέλεσης του προγράμματος, και η παρούσα Διπλωματική Εργασία σε ηλεκτρονική μορφή. Αναλυτικά τα αρχεία που περιέχει το CD-ROM είναι:

- DiplomaThesis.pdf: Η Διπλωματική Εργασία σε ηλεκτρονική μορφή (PDF).

- usb_pci.c: Αρχείο πηγαίου κώδικα που περιέχει τις συναρτήσεις για την αναγνώριση των UHCI Host Controllers που είναι συνδεδεμένοι στον δίαυλο PCI ενός υπολογιστικού συστήματος.

- usblib.c: Αρχείο πηγαίου κώδικα που περιέχει τις συναρτήσεις για την επικοινωνία με τους καταχωρητές του UHCI.

- usblib.h: Το αντίστοιχο αρχείο-επικεφαλίδα του αρχείου usblib.c.

- irp_urb.c: Αρχείο πηγαίου κώδικα που περιέχει τις συναρτήσεις διαχείρισης των Pipes, IRPs, και URBs.

- irp_urb.h: Το αντίστοιχο αρχείο-επικεφαλίδα του αρχείου irp_urb.c. Εδώ ορίζονται οι δομές δεδομένων για την περιγραφή των Pipes, IRPs, και URBs, καθώς και οι δείκτες που ορίζουν τις λίστες των IRPs και URBs.

- frame_list.c: Αρχείο πηγαίου κώδικα που περιέχει τις συναρτήσεις κατασκευής του Schedule ενός frame του Host Controller, σύμφωνα με τις προδιαγραφές του UHCI.

- frame_list.h: Το αντίστοιχο αρχείο-επικεφαλίδα του αρχείου frame_list.c. Εδώ ορίζονται οι δομές δεδομένων που θα χρησιμοποιηθούν για το Scheduling του UHCI.

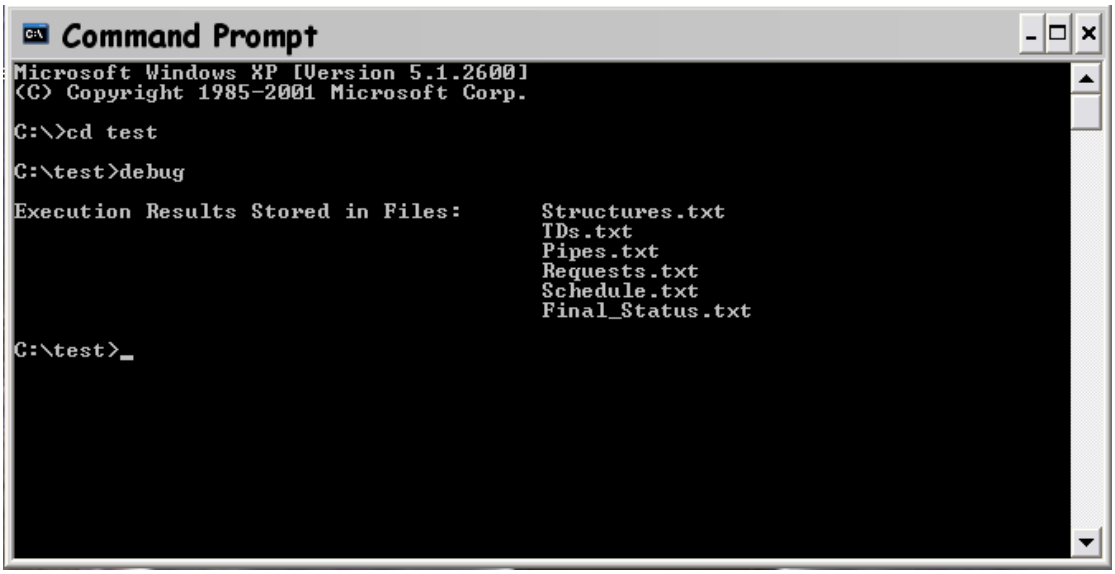
- schedule.c: Αρχείο πηγαίου κώδικα που περιέχει τις συναρτήσεις χρονοδρομολόγησης, και επιστροφής του status των transactions ενός Host Controller.

- debug.c: Αρχείο πηγαίου κώδικα που πραγματοποιεί τον έλεγχο της ορθότητας του πηγαίου κώδικα των παραπάνω αρχείων χρησιμοποιώντας έναν εικονικό Host Controller, προσομοιώνοντας κατά το δυνατόν την λειτουργία ενός πραγματικού Host Controller.

- debug.h: Το αντίστοιχο αρχείο-επικεφαλίδα του αρχείου debug.c.

- sample.c: Αρχείο πηγαίου κώδικα όπου ορίζονται Pipes και IRPs για την πραγματοποίηση ενός πειραματικού Scheduling, με σκοπό τον έλεγχο της ορθότητας του πηγαίου κώδικα.

- debug.exe: Το εκτελέσιμο αρχείο που παράγεται από την μετάφραση των παραπάνω αρχείων πηγαίου κώδικα. Τα αποτελέσματα της εκτέλεσής του αποθηκεύονται σε αρχεία κειμένου. Στην οθόνη τυπώνεται απλώς ένα μήνυμα που ενημερώνει για τα αρχεία αυτά.



Σχήμα Δ.1: Αποτέλεσμα Εκτέλεσης του Εκτελέσιμου Αρχείου

- Pipes.txt: Στο αρχείο αυτό τυπώνονται τα χαρακτηριστικά των πειραματικών Pipes που ορίζονται στο αρχείο sample.c.

- Requests.txt: Στο αρχείο αυτό τυπώνονται τα χαρακτηριστικά των πειραματικών IRP που ορίζονται στο αρχείο sample.c. Αρχικά παρουσιάζεται η σειρά με την οποία λαμβάνονται τα IRPs από τις λίστες IRP των Pipes, έτσι ώστε να διασφαλιστεί η δικαιοσύνη ανάμεσα στα Pipes. Στη συνέχεια παρουσιάζεται η αντιστοίχιση κάθε IRP στα αντίστοιχα URB, υπολογίζοντας για κάθε URB τον υπολογιζόμενο χρόνο διαύλου του transaction που αντιπροσωπεύει.

- Structures.txt: Στο αρχείο αυτό τυπώνεται η διεύθυνση βάσης του Frame List (*Frame List Base Address*) για το Frame List που δεσμεύεται για τον εικονικό Host Controller του πειράματος. Στη συνέχεια τυπώνεται η διεύθυνση των Queue Heads που δεσμεύονται για την πραγματοποίηση του Scheduling του εικονικού Host Controller. Οι παραπάνω διευθύνσεις είναι χρήσιμες για την επαλήθευση της ορθότητας της υλοποίησης του Scheduling.

- TDs.txt: Στο αρχείο αυτό τυπώνονται οι διευθύνσεις των Transfer Descriptors που δεσμεύονται γαι την υλοποίηση του Scheduling για κάθε Frame του εικονικού Host Controller, για την επαλήθευση της ορθότητας της υλοποίησης του Scheduling.

- Schedule.txt: Στο αρχείο αυτό ο αναγνώστης μπορεί να παρακολουθήσει την εξέλιξη του Scheduling για κάθε frame του εικονικού Host Controller. Πριν την χρονοδρομολόγηση κάθε frame, τυπώνονται τα περιεχόμενα των λιστών URB του εικονικού Host Controller. Στη συνέχεια τυπώνεται το ποσοστό του χρόνου διαύλου που παραχωρήθηκε για κάθε τύπο μεταφοράς δεδομένων. Ακολουθούν τα περιεχόμενα όλων των δομών δεδομένων του Frame (Frame List Entry, Queue Heads, Transfer Descriptors) που χρησιμοποιήθηκαν. Τέλος, τυπώνεται το εικονικό Status που ανατέθηκε σε κάθε Transfer

Descriptor. Συγκρίνοντας τις εκτυπώσεις του προγράμματος για κάθε frame, ο αναγνώστης μπορεί εύκολα να παρατηρήσει την αναδόμηση των λιστών URB του συστήματος, μέχρι να ολοκληρωθεί η διαδικασία χρονοδρομολογώντας όλες τις αιτήσεις μεταφοράς δεδομένων.

- Final_Status.txt: Στο αρχείο αυτό τυπώνεται το τελικό Status κάθε IRP μετά την ολοκλήρωση της χρονοδρομολόγησης των αιτήσεων μεταφοράς δεδομένων και την εικονική εκτέλεση όλων των transactions.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]. Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, NEC Corporation (Σεπτέμβριος 1998). Universal Serial Bus Specification, Revision 1.1. Διαθέσιμο στην:
http://www.usb.org/developers/docs/usbspec.zip

[2]. Intel Corporation (Μάρτιος 1996). Universal Host Controller Interface (UHCI) Design Guide, Revision 1.1. Διαθέσιμο στην:
ftp://download.intel.com/technology/usb/UHCI11D.pdf

[3]. Compaq Computer Corporation, Microsoft Corporation, National Semiconductor Corporation (Σεπτέμβριος 1999). Open Host Controller Interface Specification for USB. Διαθέσιμο στην:
ftp://ftp.compaq.com/pub/supportinformation/papers/hcir1_0a.pdf

[4]. Intel Corporation (Μάρτιος 2002). Enhanced Host Controller Inteface Specification for Universal Serial Bus. Διαθέσιμο στην:
http://www.intel.com/technology/usb/download/ehci-r10.pdf

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

CRC	Cyclic Redundancy Check
CPU	Central Processing Unit
DMA	Direct Memory Access
DWord	Double Word
EOF	End of Frame
EOP	End of Packet
HC	Host Controller
HCD	Host Controller Driver
HCDI	Host Controller Driver Interface
IRP	Input/Output Request Packet
LSb	Least Significant Bit
MSb	Most Significant Bit
OpenHCI	Open Host Controller Interface
PC	Personal Computer
PCI	Peripheral Component Interconnect
PID	Packet Identification
QH	Queue Head
SOF	Start of Frame
SOP	Start of Packet
SW	Software
TD	Transfer Descriptor
UHCI	Universal Host Controller Interface
URB	Universal Serial Bus Request Block
USB	Universal Serial Bus
USBD	Universal Serial Bus Driver